

(11)特許出願公開番号

特開平8-263290

(43)公開日 平成8年(1996)10月11日

(51) Int.Cl.⁸

G O 6 F 9/38

識別記号

3 7 0

380

庁内整理番号

FI

G O 6 F 9/38

技術表示箇所

3 7 0 B

380C

審査請求 未請求 請求項の数18 O.L (全 73 頁)

(21)出題番号

特願平7-60797

(22) 出題日

平成7年(1995)3月20日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(71)出願人 390023928

日立エンジニアリング株式会社

茨城県日立市幸町3丁目2番1号

(72)発明者 榊原 榮二

東京都小平市上水本町5丁目20番1号 株式会社日立製作所半導体事業部内

(72) 発明者 三ツ石 直幹

東京都小平市上水本町5丁目20番1号 株式会社日立製作所半導体事業部内

(74) 代理人 弁理士 秋田 収喜

最終頁に続く

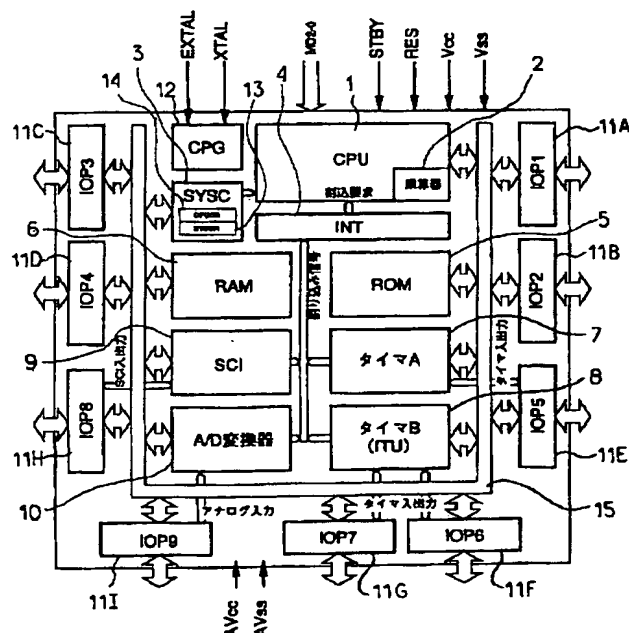
(54) 【発明の名称】 データ処理装置

(57) 【要約】

【目的】 互換性を維持しつつニーズに応じた乗算性能が得られ、しかも処理性能の向上を図ることが可能な技術を提供する。

【構成】 命令を実行する実行手段を制御する制御手段と乗算手段を設け、制御手段と乗算手段とが並列動作する第1の命令とともに、前記乗算手段が動作する第2の命令を有し、乗算手段を前記制御手段に内蔵するように構成する。第1の命令は積和命令であるとともに、第2の命令は乗算命令になっている。

图 1



BEST AVAILABLE COPY

【特許請求の範囲】

【請求項 1】 所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第 1 の命令とともに、前記乗算手段が動作する第 2 の命令を有することを特徴とするデータ処理装置。

【請求項 2】 前記第 2 の命令が第 1 の命令と同期していることを特徴とする請求項 2 に記載のデータ処理装置。

【請求項 3】 前記第 1 の命令は積和命令であるとともに、前記第 2 の命令は乗算命令であることを特徴とする請求項 1 または 2 に記載のデータ処理装置。

【請求項 4】 前記乗算手段は前記制御手段に内蔵されていることを特徴とする請求項 1 乃至 3 のいずれか 1 項に記載のデータ処理装置。

【請求項 5】 前記乗算手段は前記制御手段から独立して設けられていることを特徴とする請求項 1 乃至 3 のいずれか 1 項に記載のデータ処理装置。

【請求項 6】 前記制御手段は乗算機能を備えていることを特徴とする請求項 5 に記載のデータ処理装置。

【請求項 7】 前記乗算手段の動作の有効／無効を指定できる手段を有することを特徴とする請求項 1 乃至 6 のいずれか 1 項に記載のデータ処理装置。

【請求項 8】 前記乗算手段の動作の有効／無効を指定できる手段をテストモードで設定することを特徴とする請求項 7 に記載のデータ処理装置。

【請求項 9】 前記乗算手段の動作の有効／無効を指定できる手段をエミュレーションモードで設定することを特徴とする請求項 7 に記載のデータ処理装置。

【請求項 10】 前記乗算手段の動作の有効／無効を指定できる手段がレジスタからなり、このレジスタはエミュレータに搭載されて、エミュレータの指定に基づいて設定することを特徴とする請求項 9 に記載のデータ処理装置。

【請求項 11】 前記乗算手段の 1 ステップ動作の実行を指定する手段を有することを特徴とする請求項を 1 乃至 10 に記載のデータ処理装置。

【請求項 12】 前記乗算手段の 1 ステップ動作の実行を指定する手段をテストモードで設定することを特徴とする請求項 11 に記載のデータ処理装置。

【請求項 13】 所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段に指定可能な複数のレジスタの組み合わせを固定にし、複数のレジスタの退避／復帰命令を有することを特徴とするデータ処理装置。

【請求項 14】 所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段に搭載されるコントロールレジスタの有効／無効を切り換える手段を有することを特徴とするデータ処

理装置。

【請求項 15】 前記コントロールレジスタの有効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行い、前記コントロールレジスタの無効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行わないことを特徴とする請求項 14 に記載のデータ処理装置。

【請求項 16】 前記コントロールレジスタの有効時には、このコントロールレジスタの設定値が有効であり、前記コントロールレジスタの無効時には、このコントロールレジスタの設定値が無視されることを特徴とする請求項 15 に記載のデータ処理装置。

【請求項 17】 所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段の搭載される固定的なスタックレジスタを設け、エミュレーションプログラムへの遷移時、エミュレーションプログラムからの復帰時に、前記固定的なスタックレジスタを用いて、ユーザが使用するスタックポインタを無視あるいは保持する手段を有することを特徴とするデータ処理装置。

【請求項 18】 前記エミュレーションプログラム実行状態で、エミュレーション遷移割り込みを禁止する手段を有することを特徴とする請求項 17 に記載のデータ処理装置

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、データ処理装置に関し、特に、半導体集積回路装置によって構成される高速かつ小型のシングルチップマイクロコンピュータに利用して有効な技術に関するものである。

【0002】

【従来の技術】半導体集積回路装置の製造技術の高度化に伴って、半導体単結晶からなるシングルチップに、中央演算処理装置（Central Processing Unit；以下、単に CPU と称する）、プログラムを格納する ROM（Read Only Memory）、書き替え可能に各種データを格納する RAM（Random Access Memory）等を含む構成素子を集積して製造した、小型のシングルチップマイクロコンピュータ（以下、単にマイクロコンピュータと称する）が広範囲に普及してきており、種々の目的のデータ処理装置として使用されてきている。このマイクロコンピュータは、CPU が同時に処理し得る情報の量によって性能が異なり、例えば 4 ビット、8 ビット、16 ビット、32 ビット等のマイクロコンピュータとして区分されている。

【0003】このようなマイクロコンピュータは、アドレス空間の拡張や、命令セットの拡大、高速化等が図られてきている。また、CPU は、ソフトウェアによって

その性能が定義されているから、前記のようにアドレス空間の拡張や、命令セットの拡大、高速化等を図ったマイクロコンピュータにおいても、既存のマイクロコンピュータのソフトウェア資産を有効に利用できることが望ましい。

【0004】このため、オブジェクトレベルで互換性を保ちつつ、アドレス空間の拡張や、命令セットの拡大、高速化等を実現した例として、例えば本出願人が先に提案した特開平 6 - 5 1 9 8 1 号公報、あるいは平成 5 年 6 月 (株) 日立製作所発行、「H 8 / 3 0 0 H シリーズ プログラミングマニュアル」等がある。

【0005】前記 CPU は、システムクロックの 2 周期である、いわゆる 2 ステートで基本命令を実行している。これに対して、1 ステートで基本命令を実行するようにし、さらに、CPU とは独立して乗算器を内蔵して高速化を図った例として、例えば平成 4 年 1 1 月日経 B P 社発行、「日経エレクトロニクス N O . 5 6 8」、P P 9 9 ~ P P 1 1 2、あるいは平成 5 年 3 月 (株) 日立製作所発行、「S H 7 0 3 2、S H 7 0 3 4 ハードウェアマニュアル」等がある。このような乗算器は積和演算と乗算に利用する。

【0006】このように高速化を図ることによって、マイクロコンピュータによって制御される各種機器の高速化や高性能化、あるいは、従来においては複数の半導体集積回路装置で構成していたものを、結合したりすることにより小型化を図ることができるようになる。

【0007】また、前記のような各種機器の高速化や高性能化、あるいは小型化は、アドレス空間が比較的小さく、命令セットが比較的小さい CPU あるいはマイクロコンピュータにおいても要求されるから、前記特開平 6 - 5 1 9 8 1 号公報等に記載されるアドレス空間の広い CPU と、アドレス空間の小さい CPU が存在する場合には、その双方の高速化を図ることが望ましい。

【0008】このような観点から、上位 CPU を開発し、これをベースにして下位 CPU へ展開できれば都合が良い。これによって、開発効率を向上することができる。さらに、半導体集積回路装置によって構成される CPU 自体の他に、クロスアセンブラや C コンパイラ、シミュレータ、リアルタイム O S 等の開発ツール等の開発も共通化して、開発効率を向上することが望ましい。

【0009】

【発明が解決しようとする課題】前記のようなマイクロコンピュータにおいて、乗算器は専用の資源を必要とするから、必ずしも積和演算や乗算の高速化を必要としない場合には、費用対効果の点で得策でない。また、例えば前記平成 5 年 3 月 (株) 日立製作所発行、「S H 7 0 3 2、S H 7 0 3 4 ハードウェアマニュアル」においては、乗算結果は専用のレジスタ (M A C) に得られるから、これを利用する場合には、別の命令によってそれを CPU の汎用レジスタに転送しなければならない。乗

算器を内蔵して乗算自体を高速化しても、そのように乗算結果を使用するまでの時間が長くなっては意味がない。

【0010】一方、従来の CPU との互換性を維持するためには、前記のように命令の追加は困難であり、追加する命令は最小限にしなければならない。また、演算結果等のフラグも互換性を保持する必要がある。積和演算についても、演算結果等のフラグを参照できれば使い勝手が良くなる。フラグの状態を判定して分岐する、いわゆる条件分岐命令などで演算結果を容易に判定し、処理の内容を変更することができるからである。かかるフラグには、オーバフロー (V)、ゼロ (Z)、ネガティブ (N) などがある。

【0011】本発明の目的は、互換性を維持しつつニーズに応じた乗算性能が得られ、しかも処理性能の向上を図ることが可能な技術を提供することにある。

【0012】本発明の他の目的は、制御手段に乗算手段を内蔵して互換性を維持しつつ処理の高速化を図ることが可能な技術を提供することにある。

【0013】本発明のその他の目的は、乗算手段を制御手段から独立して設け、しかも制御手段に乗算機能を備えさせることにより、製造費用の低減が可能な技術を提供することにある。

【0014】本発明の前記ならびにそのほかの目的と新規な特長は、本発明書の記述および添付図面から明らかになるであろう。

【0015】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【0016】(1) 本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第 1 の命令とともに、前記乗算手段が動作する第 2 の命令を有し、乗算手段は前記制御手段に内蔵されている。また、第 1 の命令は積和命令であるとともに、第 2 の命令は乗算命令になっている。積和命令のアドレッシングモードは、いわゆるポストインクリメントレジスタ間接とする。乗算手段には結果を判定するフラグ検出手段を設け、乗算命令時はフラグ検出結果を制御手段に供給して、保持させる手段を設ける。

【0017】(2) 本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第 1 の命令とともに、前記乗算手段が動作する第 2 の命令を有し、乗算手段は前記制御手段から独立して設けられている。また、制御手段は乗算機能を備えている。

【0018】(3) 本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段に指定可能な複数のレジスタの組み合わせを固定にし、複数のレジスタの

退避／復帰命令を有している。

【0019】(4) 本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段に搭載されるコントロールレジスタの有効／無効を切り換える手段を有し、コントロールレジスタの有効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行い、前記コントロールレジスタの有効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行わない。

【0020】(5) 本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段の搭載される固定的なスタックレジスタを設け、エミュレーションプログラムへの遷移時、エミュレーションプログラムからの復帰時に、前記固定的なスタックレジスタを用いて、ユーザが使用するスタックポインタを無視あるいは保持するかを指定する手段を有している。

【0021】

【作用】上記した(1)の手段によれば、乗算器(乗算手段)を内蔵することによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。また、ポストインクリメントレジスタ間接により、多数のデータの積和演算を連続して処理することができる。さらに、乗算の結果(積、フラグ)を直ちに利用できるから、実質的な乗算の実行速度を向上することができる。

【0022】乗算器とCPU(制御手段)を一体に構成して、乗算器・CPU間の配線を短縮して、物理的規模を縮小する。また、高速化に寄与することができる。

【0023】上記した(2)の手段によれば、乗算器を取外し可能に(独立して)設けることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位CPUを実現し、論理的・物理的規模を縮小し、製造費用を低減した別のマイクロコンピュータを容易に開発することができる。また、乗算器を取外したCPUにおいても、汎用的な乗算命令をサポートすることによって、使い勝手の低下を防止できる。さらに、乗算器を使用するか使用しないかの制御信号(有効／無効)を与えて制御することによって、テスト性を向上したり、エミュレータを共通化したりすることができる。さらにまた、全体的な開発効率を向上することができる。

【0024】乗算器を削除した場合、乗算は除算と同一のシーケンスで実行できる。積和演算はサポートせず、積和演算の特殊なシーケンスをサポートしないことによって論理規模の縮小を更に行うことができる。テスト命令をサポートすることによって、論理規模の増加を最低限にして、テストの容易性を向上することができる。

【0025】上記した(3)の手段によれば、複数レジスタの退避／復帰命令を持ち、この組み合わせを固定的に

することによって、論理規模の縮小を図ることができ、また、高速化を図ることができる。レジスタの本数の異なる命令を複数命令サポートすることによって、使い勝手の低下を防ぐことができる。

【0026】さらに、内部動作のパイプラインに対応して、入出力タイミングの異なるレジスタ選択回路を複数持つことにより、レジスタ間演算命令などの基本命令を実質的に1命令／1ステート実行を行うことができる。

【0027】上記した(4)の手段によれば、コントロールレジスタの有効／無効を切り換えることで、スタックの節約と、割込み応答時間の高速化に寄与することができる。また、互換性を維持することができる。

【0028】上記した(5)の手段によれば、エミュレータ専用の固定スタックポインタを持つことにより、エミュレータのサポートを容易にすることができる。また、論理規模の増加を最低限にして、エミュレータの設計を容易にすることができる。エミュレータ専用スタックポインタの一部のアドレスを、CPU外部から与えるようにして、スタックレジスタをリロケータブルにし、マイクロコンピュータのアドレス配置などに容易に対応することができる。

【0029】以下、本発明について、図面を参照して実施例とともに詳細に説明する。

【0030】なお、実施例を説明するための全図において、同一機能を有するものは同一符号を付け、その繰り返しの説明は省略する。

【0031】

【実施例】図1に、本発明の適用されたデータ処理装置の一例であるシングルチップマイクロコンピュータ(以下、単にマイクロコンピュータと称する)のブロック図を示す。マイクロコンピュータは、CPU1、乗算器2、システムコントローラ(SYSC)3、割込コントローラ(INT)4、ROM5、RAM6、タイマA7、タイマB8、シリアルコミュニケーションインタフェース(SCI)9、A/D変換器10、第1乃至第9入出力ポート(IOP1～IOP9)11A～11I、クロック発振器(CPG)12の機能ブロック乃至はモジュールから構成され、公知の半導体製造技術により1つの半導体基板上に半導体集積回路装置として形成される。CPU1は、乗算器2を内蔵してなる。システムコントローラ(SYSC)3は、システムコントロールレジスタ(SYSCR)13および制御レジスタ(CPUCR)14を内蔵している。

【0032】かかるマイクロコンピュータは、電源端子として、グラウンドレベル(Vss)、電源電圧レベル(Vcc)、その他専用制御端子として、リセット(RESET)、スタンバイ(STBY)、モード制御(MD0～2)、クロック入力(EXTAL、XTAL)端子を有する。クロック入力(EXTAL、XTAL)端子に接続される、図示はされない水晶振動子に基づいて、ク

ロック発振器が生成するシステムクロック ($\phi 1$ 、 $\phi 2$) に同期して、マイクロコンピュータは動作する。或は外部クロックをEX T A L端子に入力してもよい。システムクロックの1周期を1ステートと呼ぶ。

【0033】これらの機能ブロックは、内部バスによって相互に接続される。内部バスは内部アドレスバス (P A B) ・内部データバス (P D B) の他、リード信号・ライト信号を含み、さらにバスサイズ信号或いはシステムクロック ($\phi 1$ 、 $\phi 2$) などを含む。

【0034】入出力ポートは、外部バス信号、入出力回路の入出力信号と兼用とされている。これらは、動作モードあるいはソフトウェアの設定により、機能を選択されて、使用される。I O P 1～3はアドレスバス出力、I O P 4、5はデータバス入出力、I O P 6はバス制御信号入出力信号と兼用とされている。外部アドレスは、それぞれ、これらの入出力ポートに含まれるバッファ回路を介して内部アドレスバスと接続されている。

【0035】内部バスおよび外部バス共に16ビットバス幅とし、バイトサイズ (8ビット) およびワードサイズ (16ビット) のリード/ライトを可能にする。なお、内部バスおよび外部バスのいずれも8ビット幅とすることもできる。バス制御信号入出力信号には、アドレ

スストロープ信号A S、リード信号R D、ライト信号H W R・L W R、ウェイト信号W A I T、エリア0選択信号C S 0などがある。割込信号は、タイマ・S C I・I O P 8から要求され、割込コントローラ (I N T) が調停して、C P Uに割込を要求する。このとき、C P Uに対し、割込要求信号とベクタ番号を与える。

【0036】R E S端子にリセット信号が加えられると、モード端子 (M D 0～2) で与えられる動作モードを取り込み、マイクロコンピュータはリセット状態になる。モード端子で設定する動作モードは、シングルチップ/拡張、アドレス空間、内蔵R O Mの有効/無効、データバス幅の初期値を8ビットまたは16ビットから選択する。

【0037】図2に、システムコントロールレジスタ (S Y S C R) 3の構成を示す。各ビットの内容を表1乃至表4に示す。

【0038】なお、ビット2、1:リザーブビットリードすると常に"0"が読み出される。ライトは無効である。

【0039】

【表1】

表1

ビット7:MACサチュレーション (M A C S)

MAC命令の飽和演算、非飽和演算を選択します。

ビット7	説明
M A C S	
0	MAC命令は非飽和演算 (初期値)
1	MAC命令は飽和演算

【0040】

【表2】

表2

ビット5、4:割込み制御モード1、0 (I N T M 1、0)

割込みコントローラの割込み制御モードを選択します。

ビット5	ビット4	割込み制御モード	説明
I N T M 1	I N T M 0		
0	0	0	1ビットで、割込みを制御します。(初期値)
0	1	1	I、U1ビットで、割込みを制御します。
1	0	2	I2～I0ビットで、割込みを制御します。
1	1	3	I、U1、I2～I0ビットで、割込みを制御します。

【0041】

【表3】

表 3

ビット 3 : NMI エッジセレクト (NMIEG)

NMI 割込みの入力エッジ選択を行ないます。

ビット 3	説明
NMIEG	
0	NMI 入力 の 立下がりエッジで割込み要求を発生 (初期値)
1	NMI 入力 の 立上がりエッジで割込み要求を発生

【 0 0 4 2 】

【 表 4 】

表 4

ビット 0 : RAM イネーブル

内蔵 RAM の有効または無効を選択します。RAME ビットはリセット状態の解除時に初期化されます。

ソフトウェアスタンバイモードでは初期化されません。

ビット 0	説明
RAME	
0	内蔵 RAM 無効
1	内蔵 RAM 有効 (初期値)

【 0 0 4 3 】 以下に、表 5 に CPU 1 の命令セットを示す。本実施例に用いられる CPU 1 の命令は合計で 7 1 種類ある。表 6 に命令とアドレッシングモードとの組合せを示す。表 7 に以下の各表に使用される記号 (オペレ

ーションの記号) の意味を示す。表 8 乃至表 1 5 に各命令の機能別一覧表を示す。

【 0 0 4 4 】

【 表 5 】

表5

命令の分類

分類	命令	サイズ	種類
データ 転送命令	MOV	BWL	7
	POP,PUSH	WL	
	LDM, STM	L	
	MOVFPE,MOVTPE	B	
算術演算命令	ADD,SUB,CMP,NEG	BWL	23
	ADDX,SUBX,DAA,DAS	B	
	INC,DEC	BWL	
	ADD,SUBS	L	
	MULXU,DIVXU,MULXS,DIVXS	BW	
	EXTU,EXTS	WL	
	TAS	B	
	MAC,LDMAC,STMAC,CLRMAC	-	
論理演算命令	AND,OR,XOR,NOT	BWL	4
シフト命令	SHAL,SHAR,SHLL,SHLR,ROTL,ROTR,ROTXL,ROTXR	BWL	8
ビット操作命令	BSET,BCLR,BNOT,BTST,BLD,BILD,BST,BIST BAND,BIAND,BOR,BIOR,BXOR,BIXOR	B	14
分岐命令	Bcc,JMP,BSR,JSR,RTS	-	5
システム制御命令	TRAPA,RTE,SLEEP,LDC,STC,ANDC,ORC,XORC,NOP	-	9
ブロック転送命令	EEPMOV	-	1

合計 71 種類

B: バイトサイズ W: ワードサイズ L: ロングワードサイズ

*1 POP.W Rn,PUSH.W Rnは、それぞれMOV.W @SP+,Rn,MOV.W Rn,@-SPと同一です。

また、POP.L ERn,PUSH.L ERnは、それぞれMOV.L @SP+,ERn,MOV.L ERn,@-SPと同一です。

*2 Bccは条件分岐命令の総称です。

30

【0045】

【表6】

表6

命令とアドレッシングモードの組合せ

機能	命令	アドレッシングモード												
		#XX	Rn	@ERn	@(d:16,ERn)	@(d:32,ERn)	@-ERn@ERn+	@a0:5	@a0:16	@a0:24	@a0:32	@(d:8,PC)	@(d:16,PC)	@-a0:8
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	-	BWL	-	-	-
	POP, PUSH	-	-	-	-	-	-	-	-	-	-	-	-	WL
	LDM, STM	-	-	-	-	-	-	-	-	-	-	-	-	L
	MOVEPE, MOVTP	-	-	-	-	-	-	-	B	-	-	-	-	-
算術演算命令	ADD, CMP	BWL	BWL	-	-	-	-	-	-	-	-	-	-	-
	SUB	WL	BWL	-	-	-	-	-	-	-	-	-	-	-
	ADDX, SUBX	B	B	-	-	-	-	-	-	-	-	-	-	-
	ADDS, SUBS	-	L	-	-	-	-	-	-	-	-	-	-	-
	INC, DEC	WL	BWL	-	-	-	-	-	-	-	-	-	-	-
	DAA, DAS	-	B	-	-	-	-	-	-	-	-	-	-	-
	MULXU, DIVXU	-	BW	-	-	-	-	-	-	-	-	-	-	-
	MULXS, DIVXS	-	BW	-	-	-	-	-	-	-	-	-	-	-
	NEG	-	BWL	-	-	-	-	-	-	-	-	-	-	-
	EXTU, EXTS	-	WL	-	-	-	-	-	-	-	-	-	-	-
	TAS	-	-	B	-	-	-	-	-	-	-	-	-	-
	MAC	-	-	-	-	-	○	-	-	-	-	-	-	-
	CLRMAC	-	-	-	-	-	-	-	-	-	-	-	-	○
	LDMAC, STMAC	-	L	-	-	-	-	-	-	-	-	-	-	-
論理演算命令	AND, OR, XOR	BWL	BWL	-	-	-	-	-	-	-	-	-	-	-
	NOT	-	BWL	-	-	-	-	-	-	-	-	-	-	-
シフト命令		-	BWL	-	-	-	-	-	-	-	-	-	-	-
ビット操作命令		-	B	B	-	-	-	B	B	-	B	-	-	-
分岐命令	Bcc, BSR	-	-	-	-	-	-	-	-	-	-	○	○	-
	JMP, JSR	-	-	-	-	-	-	-	-	○	-	-	-	○
	RTS	-	-	-	-	-	-	-	-	-	-	-	-	○
システム制御命令	TRAPA	-	-	-	-	-	-	-	-	-	-	-	-	○
	RTE	-	-	-	-	-	-	-	-	-	-	-	-	○
	SLEEP	-	-	-	-	-	-	-	-	-	-	-	-	○
	LDC	B	B	W	W	W	W	-	W	-	W	-	-	-
	STC	-	B	W	W	W	W	-	W	-	W	-	-	-
	ANDC, ORC, XORC	B	-	-	-	-	-	-	-	-	-	-	-	-
	NOP	-	-	-	-	-	-	-	-	-	-	-	-	○
ブロック転送命令		-	-	-	-	-	-	-	-	-	-	-	-	BW

【0046】

【表7】

表7

《オペレーションの記号》

Rd	汎用レジスタ(デスティネーション側)*
Rs	汎用レジスタ(ソース側)*
Rn	汎用レジスタ*
ERn	汎用レジスタ (32ビットレジスタ)
MAC	積和レジスタ (32ビットレジスタ)
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
EXR	エクステンドレジスタ
CCR	コンディションコードレジスタ
N	CCRのN (ネガティブ) フラグ
Z	CCRのZ (ゼロ) フラグ
V	CCRのV (オーバフロー) フラグ
C	CCRのC (キャリ) フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
+	加算
-	減算
×	乗算
÷	除算
^	論理積
∨	論理和
⊕	排他的論理和
→	転送
~	反転論理(論理的補数)
:3/:8/:16/:24/:32	3/8/16/24/32ビット長

- * 汎用レジスタは、8ビット (R0H~R7H, R0L~R7L)、16ビット (R0~R7, E0~E7)、または32ビットレジスタ (ER0~ER7) です。

30

【0047】

【表8】

40

表 8

命令の機能別一覧(1)

分類	命令	サイズ*	機 能
データ転送命令	MOV	B/W/L	(EAs)→Rd, Rs→(EAd) 汎用レジスタと汎用レジスタ、または汎用レジスタとメモリ間でデータ転送します。また、イミディエイトデータを汎用レジスタに転送します。
	MOVFP	B	(EAs)→Rd 外部メモリの内容(@aa:16で指定)をEクロックに同期したタイミングで汎用レジスタに転送します。
	MOVTP	B	Rs→(EAs) 汎用レジスタの内容をEクロックに同期したタイミングで外部メモリ(@aa:16で指定)に転送します。
	POP	W/L	@SP+→Rn スタックから汎用レジスタへデータを復帰します。 POP.W RnはMOV.W @SP+,Rnと、また、 POP.L ERnはMOV.L @SP+,ERnと同一です。
	PUSH	W/L	Rn→@-SP 汎用レジスタの内容をスタックに退避します。 PUSH.W RnはMOV.W Rn,@-SPと、また、 PUSH.L ERnはMOV.L ERn,@-SPと同一です。
	LDM	L	@SP+→Rn (レジスタ群) スタックから複数の汎用レジスタへデータを復帰します。
	STM	L	Rn (レジスタ群) →@-SP 複数の汎用レジスタの内容をスタックに退避します。

【 0 0 4 8 】

【 表 9 】

表9

命令の機能別一覧(2)

分類	命令	サイズ*	機能
命令 機能別 一覧	ADD SUB	B/W/L	$Rd \pm Rs \rightarrow Rd, Rd \pm \#IMM \rightarrow Rd$ 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の加減算を行います（バイトサイズでの汎用レジスタとイミディエイトデータ間の減算はできません。SUBX命令またはADD命令を使用してください）。
	ADDX SUBX	B	$Rd \pm Rs \pm C \rightarrow Rd, Rd \pm \#IMM \pm C \rightarrow Rd$ 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間のキャリ付きの加減算を行います。
	INC DEC	B/W/L	$Rd \pm 1 \rightarrow Rd, Rd \pm 2 \rightarrow Rd$ 汎用レジスタに1または2を加減算します（バイトサイズで1の加減算のみ可能です）。
	ADDS SUBS	L	$Rd \pm 1 \rightarrow Rd, Rd \pm 2 \rightarrow Rd, Rd \pm 4 \rightarrow Rd$ 32ビットレジスタに1、2、または4を加減算します。
	DAA DAS	B	$Rd(10進補正) \rightarrow Rd$ 汎用レジスタ上の加減算結果をCCRを参照して4ビットBCDデータに補正します。
	MULXU	B/W	$Rd \times Rs \rightarrow Rd$ 汎用レジスタと汎用レジスタ間の符号なし乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗算が可能です。
	MULXS	B/W	$Rd \times Rs \rightarrow Rd$ 汎用レジスタと汎用レジスタ間の符号付き乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗算が可能です。
	DIVXU	B/W	$Rd \div Rs \rightarrow Rd$ 汎用レジスタと汎用レジスタ間の符号なし除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	DIVXS	B/W	$Rd \div Rs \rightarrow Rd$ 汎用レジスタと汎用レジスタ間の符号付き除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	CMP	B/W/L	$Rd - Rs, Rd - \#IMM$ 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の比較を行い、その結果をCCRに反映します。
	NEG	B/W/L	$0 - Rd \rightarrow Rd$ 汎用レジスタの内容の2の補数(算術的補数)をとります。
	EXTU	W/L	$Rd(ゼロ拡張) \rightarrow Rd$ 16ビットレジスタの下位8ビットをワードサイズにゼロ拡張します。 または、32ビットレジスタの下位16ビットをロングワードサイズにゼロ拡張します。

【0049】

【表10】

表 10

命令の機能別一覧(3)

分類	命令	サイズ*	機 能
算術演算命令	EXTS	W/L	Rd(符号拡張)→Rd 16ビットレジスタの下位8ビットをワードサイズに符号拡張します。 または、32ビットレジスタの下位16ビットをロングワードサイズに符号拡張します。
	TAS	B	@ERd-0、(1)→(<ビット7>of @ERd) メモリの内容をテストした後、最上位ビット(ビット7)を"1"にセットします。
	MAC	—	(EAs)×(EAd)+MAC→MAC メモリとメモリ間の符号付き乗算を行い、結果を積和レジスタに加算します。 16ビット×16ビット+32ビット→32ビットの飽和演算、 16ビット×16ビット+42ビット→42ビットの非飽和演算が可能です。
	CLRMAC	—	0→MAC 積和レジスタをゼロクリアします。
	LDMAC STMAC	L	Rs→MAC、MAC→Rd 汎用レジスタと積和レジスタ間でデータ転送します。
論理演算命令	AND	B/W/L	Rd∧Rs→Rd、Rd∧#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理積をとります。
	OR	B/W/L	Rd∨Rs→Rd、Rd∨#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理和をとります。
	XOR	B/W/L	Rd⊕Rs→Rd、Rd⊕#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の排他的論理和をとります。
	NOT	B/W/L	~Rd→Rd 汎用レジスタの内容の1の補数(論理的補数)をとります。
シフト命令	SHAL SHAR	B/W/L	Rd(シフト処理)→Rd 汎用レジスタの内容を算術的にシフトします。
	SHLL SHLR	B/W/L	Rd(シフト処理)→Rd 汎用レジスタの内容を論理的にシフトします。
	ROTL ROTR	B/W/L	Rd(ローテート処理)→Rd 汎用レジスタの内容をローテートします。
	ROTXL ROTXR	B/W/L	Rd(ローテート処理)→Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。

48

【0050】

【表11】

表 1 1

命令の機能別一覧(4)

分類	命令	サイズ*	機 能
命令機能別一覧	BSET	B	1→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを"1"にセットします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
	BCLR	B	0→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを"0"にクリアします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
	BNOT	B	~(<ビット番号>of<EAd>)→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定されます。
	BTST	B	~(<ビット番号>of<EAd>)→Z 汎用レジスタまたはメモリのオペランドの指定された1ビットをテストし、ゼロフラグに反映します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定されます。
	BAND	B	C^(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定され1ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。
	BIAND	B	C^ [~(<ビット番号>of<EAd>)] →C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理積をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BOR	B	Cv(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定され1ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。
	BIOR	B	Cv [~(<ビット番号>of<EAd>)] →C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。

【0051】

【表 1 2】

表 1 2

命令の機能別一覧(5)

分類	命令	サイズ*	機 能
4P 命令 組	BXOR	B	$C \oplus (<\text{ビット番号}> \text{of } <\text{EAd}>) \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定され1ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。
	BIXOR	B	$C \oplus [\sim(<\text{ビット番号}> \text{of } <\text{EAd}>)] \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BLD	B	$(<\text{ビット番号}> \text{of } <\text{EAd}>) \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットをキャリフラグに転送します。
	BILD	B	$\sim(<\text{ビット番号}> \text{of } <\text{EAd}>) \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグに転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BST	B	$C \rightarrow (<\text{ビット番号}> \text{of } <\text{EAd}>)$ 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグの内容を転送します。
	BIST	B	$\sim C \rightarrow (<\text{ビット番号}> \text{of } <\text{EAd}>)$ 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグを反転して転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。

【0052】

【表13】

表 1 3

命令の機能別一覧(6)

分類	命令	サイズ*	機 能																																																			
分岐命令	Bcc	—	<p>指定した条件が成立しているとき、指定されたアドレスへ分岐します。 分岐条件を下表に示します。</p> <table border="1"> <thead> <tr> <th>ニーモニック</th><th>説明</th><th>分岐条件</th></tr> </thead> <tbody> <tr> <td>BRA(BT)</td><td>Always(True)</td><td>Always</td></tr> <tr> <td>BRN(BF)</td><td>Never(False)</td><td>Never</td></tr> <tr> <td>BHI</td><td>High</td><td>$CVZ=0$</td></tr> <tr> <td>BLS</td><td>Low or Same</td><td>$CVZ=1$</td></tr> <tr> <td>BCC(BHS)</td><td>Carry Clear(High or Same)</td><td>$C=0$</td></tr> <tr> <td>BCS(BLO)</td><td>Carry Set(Low)</td><td>$C=1$</td></tr> <tr> <td>BNE</td><td>Not Equal</td><td>$Z=0$</td></tr> <tr> <td>BEQ</td><td>Equal</td><td>$Z=1$</td></tr> <tr> <td>BVC</td><td>oVerflow Clear</td><td>$V=0$</td></tr> <tr> <td>BVS</td><td>oVerflow Set</td><td>$V=1$</td></tr> <tr> <td>BPL</td><td>PLus</td><td>$N=0$</td></tr> <tr> <td>BMI</td><td>MInus</td><td>$N=1$</td></tr> <tr> <td>BGE</td><td>Greater or Equal</td><td>$N\oplus V=0$</td></tr> <tr> <td>BLT</td><td>Less Than</td><td>$N\oplus V=1$</td></tr> <tr> <td>BGT</td><td>Greater Than</td><td>$ZV(N\oplus V)=0$</td></tr> <tr> <td>BLE</td><td>Less or Equal</td><td>$ZV(N\oplus V)=1$</td></tr> </tbody> </table>	ニーモニック	説明	分岐条件	BRA(BT)	Always(True)	Always	BRN(BF)	Never(False)	Never	BHI	High	$CVZ=0$	BLS	Low or Same	$CVZ=1$	BCC(BHS)	Carry Clear(High or Same)	$C=0$	BCS(BLO)	Carry Set(Low)	$C=1$	BNE	Not Equal	$Z=0$	BEQ	Equal	$Z=1$	BVC	oVerflow Clear	$V=0$	BVS	oVerflow Set	$V=1$	BPL	PLus	$N=0$	BMI	MInus	$N=1$	BGE	Greater or Equal	$N\oplus V=0$	BLT	Less Than	$N\oplus V=1$	BGT	Greater Than	$ZV(N\oplus V)=0$	BLE	Less or Equal	$ZV(N\oplus V)=1$
	ニーモニック	説明	分岐条件																																																			
	BRA(BT)	Always(True)	Always																																																			
	BRN(BF)	Never(False)	Never																																																			
	BHI	High	$CVZ=0$																																																			
	BLS	Low or Same	$CVZ=1$																																																			
	BCC(BHS)	Carry Clear(High or Same)	$C=0$																																																			
	BCS(BLO)	Carry Set(Low)	$C=1$																																																			
	BNE	Not Equal	$Z=0$																																																			
	BEQ	Equal	$Z=1$																																																			
	BVC	oVerflow Clear	$V=0$																																																			
	BVS	oVerflow Set	$V=1$																																																			
	BPL	PLus	$N=0$																																																			
	BMI	MInus	$N=1$																																																			
	BGE	Greater or Equal	$N\oplus V=0$																																																			
	BLT	Less Than	$N\oplus V=1$																																																			
	BGT	Greater Than	$ZV(N\oplus V)=0$																																																			
	BLE	Less or Equal	$ZV(N\oplus V)=1$																																																			
JMP	—	指定されたアドレスへ無条件に分岐します。																																																				
BSR	—	指定されたアドレスへサブルーチン分岐します。																																																				
JSR	—	指定されたアドレスへサブルーチン分岐します。																																																				
RTS	—	サブルーチンから復帰します。																																																				

【 0 0 5 3 】

【 表 1 4 】

表14

命令の機能別一覧(7)

分類	命令	サイズ*	機 能
システム制御命令	TRAPA	—	命令トラップ例外処理を行います。
	RTE	—	例外処理ルーチンから復帰します。
	SLEEP	—	低消費電力状態に移移します。
	LDC	B/W	(EAs)→CCR、(EAs)→EXR 汎用レジスタまたはメモリの内容をCCR、EXRに転送します。また、イミディエイトデータをCCR、EXRに転送します。CCR、EXRは8ビットですが、メモリとCCR、EXR間の転送はワードサイズで行われ、上位8ビットが有効になります。
	STC	B/W	CCR→(EAd)、EXR→(EAd) CCR、EXRの内容を汎用レジスタまたはメモリに転送します。CCR、EXRは8ビットですが、CCR、EXRとメモリ間の転送はワードサイズで行われ、上位8ビットが有効になります。
	ANDC	B	CCR∧#IMM→CCR、EXR∧#IMM→EXR CCR、EXRとイミディエイトデータの論理積をとります。
	ORC	B	CCR∨#IMM→CCR、EXR∨#IMM→EXR CCR、EXRとイミディエイトデータの論理和をとります。
	XORC	B	CCR⊕#IMM→CCR、EXR⊕#IMM→EXR CCR、EXRとイミディエイトデータの排他的論理和をとります。
	NOP	—	PC+2→PC PCのインクリメントだけを行います。

【0054】

【表15】

表15

命令の機能別一覧(8)

分類	命令	サイズ*	機 能
ブロック転送命令	EEPMOV.B	—	if R4L≠0 then Repeat @ER5+→@ER6+ R4L-1→R4L Until R4L=0 else next;
	EEPMOV.W	—	if R4≠0 then Repeat @ER5+→@ER6+ R4-1→R4 Until R4=0 else next; ブロック転送命令です。ER5で示されるアドレスから始まり、R4LまたはR4で指定されるバイト数のデータを、ER6で示されるアドレスのロケーションへ転送します。転送終了後、次の命令を実行します。

【0055】基本的な命令は平成5年6月(株)日立製作所発行『H8/300Hシリーズプログラミングマニュアル』などに記載のCPUと同様であり、いわゆる、ロードストアアーキテクチャを採用している。命令とア

ドレッシングモードの組み合わせを削減し、CPUの命令制御の論理規模・物理的規模を縮小できる。

【0056】本発明のCPUは、上記従来CPUに対して命令実行時間の高速化を実現している。

【0057】CPUの命令は、2バイト（ワード）を単位にしている。各命令は下記のようなオペレーションフィールド（op）、レジスタフィールド（r）、EA拡張部（EA）、およびコンディションフィールド（cc）から構成されている。

【0058】（1）オペレーションフィールド
命令の機能を表し、アドレッシングモードの指定、オペランドの処理内容を指定する。命令の先頭4ビットを必ず含んでいる。2つのオペレーションフィールドを持つ場合もある。

【0059】（2）レジスタフィールド
汎用レジスタを指定する。アドレスレジスタのとき3ビット、データレジスタのとき3ビットまたは4ビットである。2つのレジスタフィールドを持つ場合、またはレジスタフィールドを持たない場合もある。

【0060】（3）EA拡張部
イミディエイトデータ、絶対アドレスまたはディスプレイメントを指定する。8ビット、16ビット、または32ビットである。

【0061】（4）コンディションフィールド
Bcc命令の分岐条件を示す。

【0062】図3に、命令の基本フォーマットの例を示す。

【0063】図4に、マイクロコンピュータにおいて、CPU1に対し乗算器2を取外し可能に設けた概略ブロック図を示す。命令レジスタ（IR）21、命令デコーダ・制御回路（CONT）22、レジスタセクタ（RSEL）23、ライトデータバッファ（DBW）24、リードデータバッファ（DBR）25、演算器（ALU）26、演算器（INC）27、汎用レジスタ（ER0～ER7）28A～28H、エミュレータスタックポインタ（EMLSP）29、プログラムカウンタ（PC）30、コンディションコードレジスタ（CCR）31、拡張レジスタ（EXR）32、アドレスバッファ（MAB）33からなる。乗算器2なしのCPU1はこれらによって構成される。各バッファやレジスタ、演算器の各ブロックの機能は、特開平5-241826号公報に記載のCPUと概略同様である。また、乗算器2を含むCPU1は、更に、バススイッチ34、乗算器2がある。

【0064】命令デコーダ・制御回路（CONT）22には、制御信号CPUS、制御信号INTM1、その他の制御信号（割り込み要求など）が入力されている。CONT22は各部を制御するための、出力タイミングの相違する制御信号A、B、Cを出力する。

【0065】なお、図中のC1およびC2は、当該信号の同期タイミングを示す。例えば、RSEL入力1のC1はφに同期して入力が行われることを示し、RSEL入力2のC2はφ#（#は論理反転）に同期して入力が行われることを示す。また、ALU入力C1は、φの

期間に入力が行われることを示し、ALU出力C2は、φ#の期間に出力が行われることを示す。ALU26とINC27は、それぞれ動作タイミングの異なった演算器であり、それぞれ、オーバーラップしつつ演算可能である。

【0066】そのほかのレジスタなどは、φ、φ#の両方でデータを入出力可能である。GB、DB、WBの各バスはφ、φ#の両方で異なったデータを転送可能である。

10 【0067】φ、φ#は互いにノーオーバーラップの関係の2相クロックとしてもよい。

【0068】レジスタセクタ（RSEL）23には、IR21乃至CONT22から命令コードの一部（レジスタ指定フィールド）が与えられる。この供給タイミングは、レジスタ指定フィールドの位置によって相違される。RSEL23は出力タイミングの相違するレジスタ選択信号A、Bを出力する。

20 【0069】例えば、平成5年6月（株）日立製作所発行『H8/300Hシリーズプログラミングマニュアル』に記載のCPUにおいては、16ビット単位の命令コードのビット7-4が、CONT22と同時に与えられ（RSEL入力1）、ビット11-8および3-0

（RSEL入力2）が、CONT22の内容と0.5ステート遅れて与えられる。RSEL入力2の反転制御信号をRSELに与える。

30 【0070】CPU1内部のDBW24、DBR25、ALU26、INC27、ER0～ER7（28A～28H）、PC30、CCR31、VAG、ABは、GBバス、DBバス、WBバスによって相互に接続されている。

【0071】2つの演算器ALU26、INC27に対し、GB、DBバスからデータを入力し、WBバスにデータを出力する。それぞれの入出力バスの数に対応した数の内部バスとして、バス即ち配線の増加による物理的規模の増加を抑止している。

40 【0072】また、ライトデータバッファ（DBW）24は内部データバスへの出力、リードデータバッファ（DBR）25は内部データバスからの入力、アドレスバッファ（MAB）33は内部アドレスバスへの出力、命令レジスタは内部データバスからの入力が可能であり、それぞれ内部バスに接続されている。ライトデータバッファ（DBW）24およびリードデータバッファ（DBR）25は32ビット構成とされる。ライトデータは32ビット一括してライトデータバッファ（DBW）24に書き込むことができ、所定のタイミングで、16ビットの内部データバスに出力される。また、内部データバスから読み出したデータを、リードデータバッファ（DBR）25に一旦格納して、32ビットのリードデータを一括して出力することができる。MABは+2のインクリメント機能を有する。

【0073】命令デコーダ・制御回路 (CONT) 22 が、IR21からの入力、CPUS信号、INTM1信号やその他の入力信号に基づいて、動作制御を行なう。制御回路の出力は所定のバッファを介して出力される。CONT22自身にも、ステート番号などがフィードバックされる。

【0074】アドレスバッファ (MAB) 33はインクリメント機能 (+2) を有する。ER0~ER7 (28A~28Hは) データレジスタまたはアドレスレジスタとして使用することができる。

【0075】EMLSP29は、ユーザには公開されていないリソースで、エミュレータに搭載されて動作するとき、ユーザプログラムとエミュレーションプログラムの間の遷移時のスタックポインタとして使用する。その内容を指定するために、一部の内容が、CPU外部から与えられる。

【0076】PC30は32ビットのカウンタであり、CPU1が次に実行する命令のアドレスを示している。コンデションコードレジスタ (CCR) 31は割り込みマスクビット (I)、キャリフラグ (C)、ゼロフラグ (Z)、ネガティブフラグ (N)、オーバフローフラグ (V) を含んでいる。

【0077】CPU1と乗算器2は、バススイッチ34を介して接続されている。また、バススイッチ34は内部データバスとのインタフェースも行なう。また、CPU1から乗算器2への制御信号を与える。乗算器2のステータス信号BUSYと、フラグ検出信号をCPU1に与える。TESTMODE信号を、例えば、SYSC3から与える。制御信号CPUSは、SYSCR14あるいはその他のレジスタの制御ビットの出力にしてもよいし、マイクロコンピュータの制御端子のようなもので指定してもよい。

【0078】図5に、制御信号CPUSを制御レジスタ (CPUCR) 14の制御ビットで構成した具体的な例を示す。図は1ビットの構成を示している。CPUCR14は、フリップフロップで構成される。フリップフロップにはリセット信号が与えられる。フリップフロップのクロックは内部ライト信号と、アドレスをデコードして得られるCPUCR選択信号の論理積信号とされる。データ入力はデータバスのビット8とされる。出力がCPUS信号とされる。また、クロックバッファCBF6を介して、データバスに出力される。クロックバッファCBF6のクロックは内部ライト信号とCPUCR選択信号の論理積信号とされる。

【0079】本レジスタのライトは、テストモードや、エミュレータに搭載した場合のブレイクモードなどでのみライト可能にするといふ。ブレイクモードなどについては、特開平6-150026などに記載されている。同様に、TESTMODE信号を生成することができる。同一のレジスタに配置することができる。

【0080】図6に、制御信号CPUSの設定方法の一例として、エミュレーション用プロセッサおよびエミュレータをブロック図で示す。エミュレーション用プロセッサ38は、マイクロコンピュータ部分にエミュレーション用インタフェース39を加えて構成される。エミュレーション用インタフェース39には、エミュレーション用プロセッサ専用の制御レジスタ41を有する。メモリ42は、ROM、RAMを含み、I/OはI/Oポート、タイマ、SCIなどを含む。

10 【0081】コネクタ部がマイクロコンピュータの代わりに応用システム (ユーザシステム) 43に装着される。エミュレーション用プロセッサ38は上記コネクタ部とインタフェースケーブル44を介し、ターゲットシステムインタフェースを用いて上記応用システム43と信号の入出力を行う。

【0082】応用システム (ユーザシステム) 43には、特に制限はされないものの、ユーザバス45が存在し、ユーザメモリ46が接続される。エミュレーション用プロセッサ38が出力し、インタフェースケーブル44を介して供給されるユーザストロブ信号に従って、ユーザメモリ46はリード/ライトされる。

20 【0083】一方、エミュレーション用プロセッサ38は上記エミュレーションインタフェース39を用いてエミュレーションバス47に接続される。エミュレーションバス47には図示はされない状態信号・制御信号などを含む。上記エミュレーションバス47を用いて、エミュレーション用プロセッサ38から、応用システム43とエミュレーション用プロセッサ38の内部状態に応じた情報などが出力され、また、エミュレーション用プロセッサ38に対し、エミュレーションのための各種制御信号が入力される。エミュレーション用プロセッサ38の、図示はされないエミュレートモード端子が電源レベルに固定され、エミュレーション用プロセッサ38内部ではエミュレートモードが設定される。

30 【0084】さらに、上記エミュレーションバス47には、特に制限はされないものの、応用システム43またはターゲットマイクロコンピュータ内蔵のメモリを代行するためのRAMでなるようなエミュレーションメモリ48がある。また、エミュレーション用プロセッサ38の制御状態やエミュレーションバス47の状態を監視して、その状態が予め設定された状態に達した時に、上記エミュレータ専用割り込みを入力して、CPUによるユーザプログラムの実行を停止させ、エミュレーション用プログラム実行状態に遷移させる (ブレイク) ためのブレイク制御回路49と、上記CPUのリード動作またはライト動作を示す信号、命令リード動作を示す信号などに基づき、エミュレーションバス47に与えられるアドレスデータさらには制御情報を逐次蓄えるリアルタイムトレース回路50などが接続される。

50 【0085】上記エミュレーションバス47が、エミュ

レーションメモリ48、ブレーク制御回路49、リアルタイムトレース回路50などに、それぞれ接続される。これらでもってマイクロコンピュータ開発装置55が構成されている。

【0086】上記エミュレーションメモリ48、ブレーク制御回路49、リアルタイムトレース回路50はコントロールバス51に接続され、コントロールバス51を介してコントロールプロセッサ52の制御を受けるようになっている。上記コントロールバス51は、エミュレーション用プロセッサ制御回路に接続されるとともに、インタフェース回路を介して、特に制限はされないもののパーソナルコンピュータなどのシステム開発装置54に接続される。例えば、システム開発装置54から入力されたプログラムをエミュレーションメモリ48に転送し、内蔵ROM上に配置されるべきかかるプログラムをCPU1がリードすると、エミュレーションメモリ48上のプログラムがリードされる。また、ブレーク条件や、リアルタイムトレース条件などもシステム開発装置54から与えることができる。

【0087】コントロールプロセッサ52は、CPUS信号をエミュレーション用プロセッサ38に供給して、乗算器の使用／不使用の選択を行うことができる。コントロールプロセッサ52は、システム開発装置54から入力された情報などに基づいて、CPUS信号を制御する。あるいは、図5のような制御レジスタを、エミュレーション用インタフェース39内に制御レジスタに設けて、エミュレータ40のソフトウェアをCPUが実行して、前記制御レジスタを指定することによって、CPUS信号を生成するようにすることができる。この場合は、エミュレーション用ソフトウェアの実行モード、いわゆるブレークモードでのみライト可能にすると都合がよい。開発途上にあるユーザのソフトウェアの誤動作によって、誤った設定を行うことがない。

【0088】エミュレーション用プロセッサ38およびエミュレータ40を複数のCPUをサポート可能にすることによって、実際のマイクロコンピュータのみを開発すればよく、開発効率を向上することができる。なお、EMLSP29のアドレス指定情報も、エミュレーション用インタフェース39内の制御レジスタで指定することができる。

【0089】エミュレーション用プロセッサ38やエミュレータ40については、特開平3-271834号公報、あるいは特開平6-150026号公報などに記載されている。

【0090】図7に、制御信号CPUS設定方法の一例である、マイクロコンピュータの主要部をブロック図で示す。CPUS信号をレジスタによらず、CMOSインバータ回路58の出力とする。かかるCMOSインバータ回路58は、Pチャネル型MOSトランジスタQ1、Nチャネル型MOSトランジスタQ2で構成される。こ

のCMOSインバータ回路58の入力は、抵抗Rを介して電源V_{dd}に接続されると共に、保護回路Q3、Q4を介して端子Pに結合される。端子Pは、ワイヤWによってグラウンドレベル電源用リードLに接続されるか、解放状態とされるかが選択され、CPUSの設定を行なう。

【0091】端子Pが解放状態とされれば、CMOSインバータ回路58の入力はハイレベルとなって、CPUS信号は非活性状態になる。一方、端子Pが、ワイヤWによって、グラウンドレベル電源用リードLに接続されれば、CMOSインバータ回路58の入力はロウレベルとなって、CPUS信号は活性状態になる。乗算器を使用可能にする。

【0092】端子Pは対応するリードを持たず、例えばプラスチックパッケージに封止された場合には、対応する端子を持たない。

【0093】これにより、半導体集積回路装置のパッケージの端子を直接利用することなく、乗算器の制御を設定できるため、一定のパッケージを用いた場合に、有効な端子数の減少防ぐことができる。この場合、端子Pをグラウンドレベル電源端子に隣接して配置すると都合がよい。

【0094】あるいは、端子Pをグラウンドレベル電源用リードLにワイヤWによって接続するか、しないかの選択を、半導体集積回路装置の配線変更として実現してもよい。CMOSインバータ回路58の入力を、半導体集積回路装置内部の電源電圧またはグラウンドのいずれに接続するかを選択すればよい。このとき、抵抗R及び端子Pは削除することができる。または、CPUSビットをPROM素子などで構成してもよい。この場合、製造者が設定を行なってもよいし、ユーザが設定を行なってもよい。

【0095】図8および図9に、CPUの内部レジスタ構成を示す。これらのレジスタは、図8の汎用レジスタおよび図9のコントロールレジスタの2つに分割される。以下、各レジスタについて説明する。

【0096】(1) 汎用レジスタ

CPUはこの汎用レジスタを8本有している。この汎用レジスタは32ビット長からなり、すべて同じ機能を有しており、アドレスレジスタとしてもデータレジスタとしても使用することができる。データレジスタとしては32ビット、16ビットおよび8ビットレジスタとして使用できる。

【0097】アドレスレジスタ及び32ビットレジスタとしては、一括して汎用レジスタER(ER0~ER7)として使用する。16ビットレジスタとしては、汎用レジスタERを分割して汎用レジスタE(E0~E7)、汎用レジスタR(R0~R7)として使用する。これらは同等の機能を有しており、16ビットレジスタを最大16本まで使用することができる。

【0098】8ビットレジスタとしては、汎用レジスタRを分割して汎用レジスタRH(R0H~R7H)、汎用レジスタRL(R0L~R7L)として使用する。これらは同等の機能を有しており、8ビットレジスタを最大16本まで使用することができる。

【0099】図10に、汎用レジスタの使用方法を示す。各レジスタは独立して使用方法を選択することができる。

【0100】汎用レジスタER7には、汎用レジスタとしての機能に加えて、スタックポインタ(SP)としての機能が割り当てられており、例外処理やサブルーチン分岐などで暗黙的に使用される。図11にスタックの状態を示す。

【0101】(2)コントロールレジスタ
コントロールレジスタは、24ビットのプログラムカウンタ(PC)と8ビットの拡張レジスタ(エクステンデッドレジスタ)(EXR)および8ビットのコンディションコードレジスタ(CCR)を含んでいる。

【0102】①プログラムカウンタ(PC)
24ビットのカウンタで、CPUが次に実行する命令のアドレスを示している。CPUの命令は、すべて2バイト(ワード)を単位としているため、最下位ビットは無効である。(命令コードのリード時には最下位ビットは"0"とみなされる)。

【0103】分岐命令の実行アドレスの上位8ビットは無視される。プログラム領域として使用できるのは、H'00000000~H'00FFFFFFFの領域である。

【0104】②拡張レジスタ(EXR)
8ビットのレジスタで、トレースビット(T)、割込みマスクビット(I2~I0)を含む8ビットで構成されている。

【0105】ビット7:トレースビット(T)
トレースビットか否かを指定する。本ビットが"0"にクリアされているときは命令を順次実行する。"1"にセットされているときは1命令実行する毎にトレース例外処理を実行する。

【0106】ビット6~4:リザーブビット
リザーブビットである。

【0107】ビット2~0:割込みマスクビット(I2~I0)

割込み要求マスクレベル(0~7)を指定する。

【0108】EXRは、LDC、STC、ANDC、ORC、XORC命令で実行することができる。このうち、STCを除く命令を実行した場合、実行終了後3ステートの間は、NMIを含めてすべての割込みは受け付けられない。

【0109】③コンディションコードレジスタ(CCR)

8ビットのレジスタで、CPUの内部状態を示す。割込

みマスクビット(I)とハーフキャリ(H)、ネガティブ(N)、ゼロ(Z)、オーバフロー(V)、キャリ(C)を含む8ビットで構成されている。

【0110】ビット7:割込みマスクビット(I)
本ビットが"1"にセットされると、割込みがマスクされる。ただし、NMIはIビットに関係なく受け付けられる。例外処理の実行が開始されたときに"1"にセットされる。

【0111】ビット6:ユーザビット/割込みマスクビット(UI)
ソフトウェア(LDC、STC、ANDC、ORC、ZORC命令)でリード/ライトできる。割込みマスクビットとしても使用可能である。

【0112】ビット5:ハーフキャリフラグ(H)
ADD. B、ADDX. B、SUB. B、SUBX. B、CMP. B、NEG. B命令の実行により、ビット3にキャリまたはボローが生じたとき"1"にセットされ、生じなかったとき"0"にクリアされる。また、ADD. W、SUB. W、CMP. W、NEG. W命令の実行により、ビット11にキャリまたはボローが生じたとき、ADD. L、SUB. L、CMP. L、NEG. L命令の実行により、ビット27にキャリまたはボローが生じたとき、"1"にセットされ、生じなかったとき"0"にクリアされる。

【0113】ビット4:ユーザビット(U)
ソフトウェア(LDC、STC、ANDC、ORC、XORC命令)でリード/ライトできる。

【0114】ビット3:ネガティブフラグ(N)
データの最上位ビットを符号ビットとみなし、最上位ビットの値を格納する。

【0115】ビット2:ゼロフラグ(Z)
データがゼロのとき"1"にセットされ、ゼロ以外のとき"0"にクリアされる。

【0116】ビット1:オーバフローフラグ(V)
算術演算命令により、オーバフローが生じたとき"1"にセットされる。それ以外のとき"0"にクリアされる。

【0117】ビット0:キャリフラグ(C)
演算の実行により、キャリが生じたとき"1"にセットされ、生じなかったとき"0"にクリアされる。キャリには次の種類がある。

【0118】(a)加算結果のキャリ

(b)減算結果のボロー

(c)シフト/ローテートのキャリ

また、キャリフラグには、ビットアキュムレータの機能があり、ビット操作命令で使用される。なお、命令によってはフラグが変化しない場合がある。CCRは、LDC、STC、ANDC、ORC、XORC命令で操作することができる。また、N、Z、V、Cの各フラグは、条件分岐命令(Bcc)で使用される。

【0119】④積和レジスタ (MAC)

64ビットのレジスタであり、積和演算結果を格納する。32ビットのMACH、MACLから構成される。MACHは下位10ビットが有効であり、上位は符号拡張されている。

【0120】図12に、CPUの基本動作タイミングを示す。

【0121】ADD. W R0、R1のようなレジスタ間演算のタイミングである。特に制限はされないものの、内部データバスは16ビットであって、内蔵ROM、RAMのリード/ライトを1ステートでリード/ライト可能とする。

【0122】T0のC2 (φ#同期。#は反転論理を示す)で、CPU1のアドレスバッファ (MAB) 33からアドレスがIABに出力される。

【0123】T1のC1 (φ同期)で、IABの内容がPABに出力され、リードサイクルが開始される。C2でリードデータが内部データバスに得られ、これをIR21にラッチする。以上の動作は以前の命令の実行の制御によって行われる。

【0124】直前の命令の実行が終了すると、最も早く命令の実行が開始される場合には、T2のC1で命令コードがCONT22に入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部 (レジスタ指定フィールド: RSEL入力信号1) がレジスタセクタ23に与えられる。

【0125】レジスタ間演算命令では、T2のC2で、PCの内容を内部バスGBに読み出して、MAB33とINC27に入力する。MAB33からアドレスIABが出力される。レジスタセクタ23に制御信号を与える。RSEL入力信号1と制御信号A (Rs-DB出力、Rd-GB出力) とに基づいて、レジスタ選択信号Bが生成される。RSEL入力信号2がレジスタセクタ23に与えられる。

【0126】T3から、次の次の命令がリードされる。T3のC1で、INC27でインクリメント (+2) された結果が、内部バスWBを経由して、PC30にライトされる。RSEL入力信号2と制御信号B (WB-Rd入力) とに基づいて、レジスタ選択信号Cが生成される。レジスタ選択信号Bがレジスタを選択して、ソース側、デスティネーション側のレジスタ (S、D) のデータをALU26に入力する。ALU26の演算内容はCONT22が制御信号Cによって指示する。加減算・論理演算・シフトなどは1クロックで演算を行うことができる。例えば、上記命令では16ビットの加算を行う。次の命令のCONT22へのロードを指示する。RSEL入力信号2と制御信号B (WB-Rd入力) とに基づいて、レジスタ選択信号Cが生成される。

【0127】T3のC2で、ALU26の演算結果

(R) が、内部バスWBを経由して、レジスタ選択信号Cが選択したデスティネーション側のレジスタにライトされる。制御信号Cによって、CCR31の更新を行う。更に次の次の命令をIR21に取り込む。同時に、次の命令の実行が開始され、例えば、PC30の内容を読み出して、MAB33とINC27に入力される。レジスタ間演算を実質的に1ステートで実行できる。2つの演算器26、27の入出力バスの数に対応した数の内部バスとして (演算器に対応して、内部バスを増加させることなく)、バス即ち配線の増加による物理的規模の増加を抑止している。

【0128】図13に、CPUの基本動作タイミングを示す。

【0129】MOV. W R0、@R1のような、レジスタ間接によるデータライトのタイミングである。

【0130】T0のC2で、CPU1のMAB33からアドレスがIABに出力される。

【0131】T1のC1で、アドレスがPABに出力され、リードサイクルが開始される。C2でリードデータが内部データバスに得られ、これをIR21にラッチする。

【0132】直前の命令の実行が終了すると、T2のC1で命令コードがCONT22に入力されて、命令の内容が解読され、各部の制御を行う。命令の一部のレジスタ指定フィールド (RSEL入力信号1) がレジスタセクタ23に与えられる。レジスタ間接によるデータライトでは、制御信号AとRSEL入力信号1とに基づいて、レジスタ選択信号Aが与えられ、アドレスとして指定されたレジスタが選択される。

【0133】T2のC2で、選択されたレジスタの内容 (A) を内部バスGBに読み出して、MAB33を経由してアドレスIABに出力される。RSEL入力信号2がレジスタセクタ23に与えられる。RSEL入力信号2と制御信号B (Rd-DB出力) とに基づいて、レジスタ選択信号Bが生成される。

【0134】T3のC1で、制御信号Cの一部がCONT22に入力され、状態遷移が行われる (ステートマシンが構成される)。IABの内容に基づいて、ライトサイクルが開始される。選択されたレジスタの内容 (D) を内部バスDBに読み出して、データバッファ (DBW) を経由して内部データバスに出力される。

【0135】T3のC2で、PC30の内容を内部バスGBに読み出して、MAB33とINC27に入力する。MAB33からアドレスIABが出力される。RSEL入力信号1と制御信号A (Rd-GB出力) とに基づいて、レジスタ選択信号Bが生成される。

【0136】T4から、次の次の命令がリードされる。

【0137】T4のC1で、INC27でインクリメント (+2) された結果が、内部バスWBを経由して、PC30にライトされる。レジスタ選択信号Bがレジスタ

を選択して、データレジスタ (D) のデータを ALU 26 に入力する。ALU 26 の演算内容は CONT 22 が制御信号 C によって指示する。転送の場合はデータのチェックのみを行う。次の命令の CONT 22 へのロードを指示する。

【0138】T4のC2で、制御信号Cによって、チェックした結果によって、CCR31の更新を行う。更に次の次の命令をIR21に取り込む。同時に、次の命令の実行が開始され、例えば、PC30の内容を読み出して、MAB33とINC27に入力される。

【0139】RSELに入力するタイミングを、RSEL入力1（アドレスレジスタ、ソースレジスタ）とRS

EL入力2（データレジスタ、ディスティネーションレジスタ）のように、レジスタ指定フィールド毎にことなつたタイミング（φ同期とφ#同期）とすることにより、命令実行の高速化を実現することができる。

【0140】表16乃至表19に、本発明に関係のある命令の説明を示す。

【0141】表16は命令コードを示し、表18は命令の実行状態を示し、表19はコンディションコードの変化を示している。表17はレジスタフィールドと汎用レジスタの対応を示している。

【0142】

【表16】

表 16

命令コード

命令	ニーゼニック	サイズ	インストラクションフォーマット									
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
CLRMAC		-	0	1	A	0						
DIVXS		B	0	1	D	0	5	1	rs	rd		
		W	0	1	D	0	5	3	rs	rd		
DIVXU		B	5	1	rs	rd						
		W	5	3	rs	rd						
LDM		L	0	1	1	0	6	D	7	rd	em+1	
		L	0	1	2	0	6	D	7	rd	em+2	
		L	0	1	3	0	6	D	7	rd	em+3	
LDMAC		L	0	3	2	ens						
		L	0	3	3	ens						
MAC		-	0	1	6	0	6	D	0	em	rd	
MULXS		B	0	1	C	0	5	0	rs	rd		
		W	0	1	C	0	5	2	rs	rd		
MULXU		B	5	0	rs	rd						
		W	5	2	rs	rd						
POP		W	6	D	7	m						
		L	0	1	0	0	6	D	7	rd	em	
PUSH		W	6	D	F	m						
		L	0	1	0	0	6	D	F	rd	em	
STM		L	0	1	1	0	6	D	F	rd	em	
		L	0	1	2	0	6	D	F	rd	em	
		L	0	1	3	0	6	D	F	rd	em	
STMAC		L	0	2	2	ens						
		L	0	2	3	ens						

<<記号説明>>

n, rd, m レジスタフィールド (4ビットで8ビットレジスタまたは16ビットレジスタを指定します。

rs, rd, mはそれぞれオペランド形式のRs, Rd, Rnに対応します。)

ens, end, em レジスタフィールド (3ビットでアドレスレジスタまたは32ビットレジスタを指定します。

ens, end, em, emはそれぞれオペランド形式のERs, ERd, ERn, ERmに対応します。)

表 1 7

アドレスレジスタ 32ビットレジスタ		16ビットレジスタ		8ビットレジスタ	
レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		:	:	:	:
		:	:	:	:
		1111	E7	1111	R7L

MOV.L ERs,@(d:32.ERd)命令の第4バイト、ビット7は"1"、"0"のどちらでも動作可能です。

【 0 1 4 4 】

【表 1 8】

表 18

命令の実行状態

命 令	1	2	3	4	5	6	7	8	9
CLRMAC	R:W NEXT	内部動作1 ステート							
DIVXS.B R _s , R _d	R:W 2nd	R:W NEXT	内部動作11 ステート						
DIVXS.W R _s , ER _d	R:W 2nd	R:W NEXT	内部動作19 ステート						
DIVXU.B R _s , R _d	R:W NEXT	内部動作11 ステート							
DIVXU.W R _s , ER _d	R:W NEXT	内部動作19 ステート							
LDM @SP ₊ , (ER _n -ER _{n+1})	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
LDM @SP ₊ , (ER _n -ER _{n+2})	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
LDM @SP ₊ , (ER _n -ER _{n+3})	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
LDMAC ER _s , MACH	R:W NEXT	内部動作11 ステート		— n 回繰り返し返す *3	—				
LDMAC ER _s , MACL	R:W NEXT	内部動作11 ステート							
MAC @ER _n , @ER _m +	R:W 2nd	R:W NEXT	R:W EAn	R:W EAm					
MULXS.B R _s , R _d	R:W 2nd	R:W NEXT	内部動作2 ステート						
MULXS.W R _s , ER _d	R:W 2nd	R:W NEXT	内部動作3 ステート						
MULXU.B R _s , R _d	R:W NEXT	内部動作2 ステート							
MULXU.W R _s , ER _d	R:W NEXT	内部動作3 ステート							
STM (ER _n -ER _{n+1}), @SP	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
STM (ER _n -ER _{n+2}), @SP	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
STM (ER _n -ER _{n+3}), @SP	R:W 2nd	R:W NEXT	内部動作11 ステート	R:W X17(H) *3	R:W X17(L) *3				
STMACH, ER _d	R:W NEXT			— n 回繰り返し返す *3	—				
STMACH, MACL, ER _d	R:W NEXT								

*3 2本退避/復帰時は2回、3本退避/復帰時は3回、4本退避/復帰時は4回繰り返し返します。

表 1 9

コンディションコードの変化

命 令	H	N	Z	V	C	定 義
CLRMAC	-	-	-	-	-	
DIVXS	-	⇔	⇔	-	-	$N = S_m \cdot \overline{D_m} + \overline{S_m} \cdot D_m$ $Z = \overline{S_m} \cdot \overline{S_{m-1}} \cdot \dots \cdot S_0$
DIVXU	-	⇔	⇔	-	-	$N = S_m$ $Z = \overline{S_m} \cdot \overline{S_{m-1}} \cdot \dots \cdot S_0$
LDM	-	-	-	-	-	
LDMAC	-	-	-	-	-	
MAC	-	-	-	-	-	
MULXS	-	⇔	⇔	-	-	$N = R_{2m}$ $Z = \overline{R_{2m}} \cdot \overline{R_{2m-1}} \cdot \dots \cdot R_0$
MULXU	-	-	-	-	-	
POP	-	⇔	⇔	0	-	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot R_0$
PUSH	-	⇔	⇔	0	-	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot R_0$
STM	-	-	-	-	-	
STMAC	-	⇔	⇔	⇔	-	N=MAC命令の結果、MACレジスタが負のとき Z=MAC命令の結果、MACレジスタが"0"のとき V=MAC命令の結果、オーバフローが発生したとき

$m = \begin{cases} 31 : \text{ロングワードサイズの時} \\ 15 : \text{ワードサイズの時} \\ 7 : \text{バイトワードサイズの時} \end{cases}$

Si : ソースオペランドのビット i
Di : ディステーションオペランドのビット i
Ri : 結果のビット i
Dn : ディステーションオペランドの指定されたビット
- : 影響なし
⇔ : 実行結果に応じて変化 (定義参照)
0 : 常に "0" クリア
1 : 常に "1" セット
* : 値を保証しません
Z' : 実行前の Z フラグ
C' : 実行前の C フラグ

【0146】積和演算を行うMAC命令、MACレジスタをクリアするCLRMAC命令、汎用レジスタの内容をMACレジスタに転送するLDMAC命令、MACレジスタの内容を汎用レジスタに転送するSTMAC命令がある。

【0147】また、汎用レジスタの待避/復帰命令には、1本のレジスタの待避/復帰命令に、PUSH、POP命令が、複数レジスタの待避/復帰命令にSTM/LDM命令がある。STM/LDM命令には、指定するレジスタ本数に対応して3種類がある。

【0148】図14に、乗算器2の概略ブロック図を示す。

【0149】乗算器2は、入力ラッチ(X)61、入力ラッチ(Y)62、部分積生成回路63、マッチプレクサ64、デコーダ65A、65B、65C、選択回路66A、66B、66C、加算器67、フィードバック回

路68、乗算結果レジスタ69などによって構成されている。

【0150】乗算器2は16×16ビットの乗算を行うことを基本動作とし、さらに、これを利用して、16×16ビット+42ビットの積和演算を可能としている。

【0151】乗算器は乗算動作は、2次のブースのデコードを用いて、16ビット×6ビットを3回行うようにされる。

【0152】16ビットの乗数Yは、 $Y = -y[16] \cdot 2^{15} + \sum (y[i] \cdot 2^{(i-1)}) = \sum (y[2j] + y[2j+1] - 2 \cdot y[2j+2]) \cdot 2^{2j}$ と表現される。i=1~15、j=0~7、y[0]=0である。

【0153】被乗数Xとの乗算は、 $X \cdot Y = \sum (y[2j] + y[2j+1] - 2 \cdot y[2j+2]) \cdot X \cdot 2^{2j}$ となる。y[2j]+y[2j+1]-2・y

$[2j+2]$ は、 $y[2j]$ 、 $y[2j+1]$ 、 $y[2j+2]$ の値の組み合わせにより、0、 ± 1 、 ± 2 の5種類があるから、部分積 $(y[2j] + y[2j+1] - 2 \cdot y[2j+2]) \cdot X$ は、0、 $\pm X$ 、 $\pm 2X$ の5種類である。この内、0、 X は直ちに得られる。 $2X$ は、1ビットの左シフト（最下位ビットは0）、 $-X$ は2の補数であり、論理反転+1で得られる。 $-2X$ は、論理反転+1の1ビットの左シフト（最下位ビットは0）で得る。

【0154】 X 側は、採りうる0、 $\pm X$ 、 $\pm 2X$ の5種類の部分積（17ビット）を生成しておく。この5種類を部分積選択回路66A～66Cに与える。

【0155】一方、 Y 入力 $y[2j]$ 、 $y[2j+1]$ 、 $y[2j+2]$ をデコードして、0、 ± 1 、 ± 2 を判定して、この結果によって、部分積選択回路66A～66Cを制御して、前記5種類の部分積を選択する。1回に2ビット単位3種類の選択を行う。これを加算器67で加算する。加算は、2ビットずつシフトしたそれぞれ17ビットの部分積を加算して、22ビット分の結果を得る。不足する上位ビットは符号拡張したデータとする。

【0156】この内、下位6ビットは、乗算結果レジスタ69のビット0～5に格納される。上位16ビットはフィードバック回路68を介して、2回めの加算に含められる。2回目の処理では、前記同様に得られた部分積選択回路66A～66Cの出力である、2ビットずつシフトしたそれぞれ17ビットの部分積と、1回めの処理の上位16ビットを加算する。22ビット分の結果を得る。不足する上位ビットは符号拡張したデータとする。この内、下位6ビットは、乗算結果レジスタ69のビット11～6に格納される。上位16ビットはフィードバック回路68を介して、2回めの加算に含められる。

【0157】同様に3回めの処理が行われる。加算結果下位20ビットが乗算結果レジスタ69のビット31～12に格納される。加算結果の最上位2ビットは無視する。

【0158】積和演算の場合は、前回の結果が同時に加算されるようにする。

【0159】更に、前回の結果の上位ビットとの4回目の加算を行って、42ビットの結果を得る。16ビット×16ビットの積和の結果を42ビットで得ることにより、約1000回の積和演算を繰り返してもオーバーフローしないことになる。

【0160】内部論理の構成上は、40ビットの結果とすれば、加算器が22ビット長でよく、論理的規模を最適化できる。

【0161】図15に、上記のワードサイズ乗算（16ビット×16ビット）の演算方法を示す。

【0162】8ビット×8ビットのバイトサイズ乗算は、上位を拡張する。符号無しの場合0拡張、符号付き

の場合符号拡張を行う。いずれの場合も、上位は全ビット”0”か全ビット”1”かのいずれかであって、ブースのデコードは0になる。このため、3回目の処理は行わずに、2回の処理で済む。

【0163】図4において、CPU1から乗算器2に、乗算を示す信号として、MUL信号（制御信号B）、符号付き／無しを示す信号としてUNSIGN信号（制御信号B）、バイト／ワードサイズを示す信号として、BYTE信号（制御信号B）、積和演算の起動信号として、MAC信号（制御信号B）、MACHからCPUへのデータ転送要求信号として、STMACH信号（制御信号C）、MACLからCPUへのデータ転送要求信号として、STMACH信号（制御信号C）、CPUからMACHへのデータ転送要求信号として、LDMACH信号（制御信号B）、CPUからMACLへのデータ転送要求信号として、LDMACH信号（制御信号B）、MACレジスタのクリア信号として、CLRMAC信号（制御信号B）、乗数の転送信号として、STX信号（制御信号B）、被乗数の転送信号として、STY信号（制御信号B）、乗算結果の転送信号として、MULRD信号（制御信号C）を与える。

【0164】また、乗算器2からCPU1へ、演算実行中を示す信号として、BUSY信号、フラグに反映すべきデータとして、VFLAG、ZFLAG、NFLAG信号が与えられる。

【0165】CPU-乗算器の相互のデータ転送にXバス、Yバスを使用する。

【0166】また、SYSCR13から飽和演算の選択を示す信号として、FIXED信号が与えられる。また、テストモード信号として、TESTMODE信号が与えられる。TESTMODE信号が活性状態になって、テストモードが指示されると、乗算器は1回の処理のみで動作を終了するようにする。

【0167】処理を短縮することによって、CPUの命令実行ステートも短縮できる。入力データの組み合わせを種々変更してテストする場合に、テスト時間を短縮できる。加算を1回しか行わないので、テスト設計を容易にすることができる。3回の処理を行って、加算結果が蓄積されて、所望の動作のテストの結果を演算結果として得にくくすることがない。

【0168】TESTMODE信号が非活性状態であっても、CPU乃至乗算器のテストを行うことができることは言うまでもない。TESTMODE信号は、前記のCPUのようにレジスタの出力として供給することができる。

【0169】表20に乗算器2の内部のフラグの検出方式およびCPU1への転送方式を示す。

【0170】

【表20】

表 20

OVF、UNFしたときの乗算器のフラグ

条件		Vフラグ	Nフラグ	Zフラグ	演算結果のMSB
OVF/UNFしないとき		0	N=結果のMSB	結果=H'00...のときZ=1、 結果<H'00...のときZ=0	MSB=Nフラグ
OVFしたとき	積和演算	1	1	結果=H'00...のときZ=1、 結果<H'00...のときZ=0	1
	飽和演算	1	0	φ	0
UNFしたとき	積和演算	1	0	結果=H'00...のときZ=1、 結果<H'00...のときZ=0	0
	飽和演算	1	1	φ	1

結果：飽和演算で、OVF/UNFしたとき、以下のようにする。

- ・OVFしたとき：演算結果=H'7FFFFFFF、N=0、Z=0にマスクする
- ・UNFしたとき：演算結果=H'80000000、N=1、Z=0にマスクする

(演算結果とフラグを一致させる)

【0171】乗算器2のフラグ仕様は次のように、①Vフラグおよび②Nフラグ、Zフラグから構成されている。

【0172】①Vフラグ

セット条件はMAC命令実行中にオーバフローまたはアンダフローが発生したときである。

【0173】クリア条件はLDMACまたはCLRMAC命令を実行したときである。

【0174】乗算器からCCRへの転送は、STMAC実行時に行われる。

【0175】従って、一連の連続した積和演算中に1回でもオーバフローまたはアンダフローが発生すると、乗算器のVフラグはセットされた状態を保持する。LDMACまたはCLRMAC命令を実行して、新しい一連の積和演算の開始が判断されると、乗算器のVフラグはクリアされる。

【0176】②Nフラグ、Zフラグ

MUL命令用N、ZフラグとMAC命令用N、Zフラグを別々に設けて出力する。

【0177】乗算器からCCRへの転送は、乗算(MUL)命令の場合、乗算結果の転送時、MAC命令の場合STMAC実行時に行われる。

【0178】なお、NフラグとZフラグは、LDMAC/CLRMACによって変化しない。

【0179】図16にVフラグ仕様の実現の概念図を示し、図17にNフラグ、Zフラグ仕様の実現の概念図を示す。

【0180】Vフラグはセットリセット型のフリップフロップ(RS-F/F)で構成され、一旦、オーバフローまたはアンダフローが発生すると、STMACにより読み出すまで状態を保持する。

【0181】N、Vフラグはラッチ回路(D-F/F)とマルチプレクサ(MPX)で構成される。MAC命令実行時の演算結果はラッチ回路に保持され、マルチプレクサに与えられる。また、乗算命令実行時の演算結果は、直接マルチプレクサに与えられる。マルチプレクサはSTMAC命令のときラッチ回路の出力を出力し、それ以外のとき演算結果を直接出力する。

【0182】MAC命令とその他の命令は並列して動作する。MAC命令のフラグを随時CCRに反映しては、並列実行中の命令のフラグ動作と矛盾してしまう。MAC命令のフラグを乗算器内部で保持して、STMAC命令実行時にCCRに転送するようにして、上記矛盾を回避することができる。

【0183】図18に、バススイッチ34のブロック図を示す。

【0184】バススイッチ34は、選択回路71A、71B、拡張回路72A、72B、72C、出力バッファ

73A、73B、73C、73Dから構成される。バススイッチは、CPU内部バスのGB、DB、WBと、乗算器のXバス、Yバスと、マイクロコンピュータの内部バスであるIDBのインタフェースを行う。

【0185】乗算の開始時、及びLDMAC命令の場合は、GB、DBからXバス、Yバスに入力される。GB、DBの入力は、選択回路71A、71Bで選択される。これは汎用レジスタ及び内部バスが32ビット構成であるために、乗数、被乗数が8ビットまたは16ビットであるために、CONT22の制御信号A及びレジスタ制御信号Aに基づいて、所定の部分が選択される。

【0186】選択回路71A、71Bの出力は、拡張回路72A、72Bに入力される。CONT22の制御信号Aに基づいて、符号無しバイトサイズ乗算(MULXU、B)の場合、上位8ビットを0拡張する。また、符号付きバイトサイズ乗算(MULXS、B)の場合、上位8ビットを符号拡張する。ワードサイズの場合は、選択回路71A、71Bの出力をそのまま出力する。

【0187】選択回路71A、71Bの出力は出力バッファ73B、73Cに入力される。CONT22の制御信号Aに基づいて、所定のタイミングで、拡張回路72A、72Bの出力をXバスまたはYバスに出力する。

【0188】乗算の終了時、及びSTMAC命令の場合は、Xバス、YバスからWBへの出力が行われる。Xバス、Yバスの入力は拡張回路72Cに入力される。これは、MACHの上位22ビットを符号拡張する。拡張回路72Cの出力は出力バッファに入力される。CONT22の制御信号Cに基づいて、所定のタイミングで、拡張回路72Cの出力をWBに出力する。

【0189】MAC命令のデータリード時、IDBからXバス、Yバスへの入力が、それぞれ1回ずつ行われる。CONT22の制御信号Aに基づいて、所定のタイミングで、IDBの内容はXバスまたはYバスに出力する。また、IDBは、DBWからの出力を入力可能とされ、DBR及びIRへデータを入力可能とされる。

【0190】図19、20に、MAC命令の動作タイミングを示す。

【0191】例えば、MAC @ER1+, @ER2+命令などの例である。この場合のER1を第1のアドレスレジスタ、ER2を第2のアドレスレジスタとする。前記同様に、T2からMAC命令の実行が開始される。

【0192】まず、プリフィックスコードの実行を行い、PC30の内容をアドレスとした命令のリードを行い、また、PC30の内容のインクリメントを行う。

【0193】T3のφ#で、レジスタ制御信号Aに基づいて、第1のアドレスの内容をGBに読み出して、MAB33に転送し、IABに出力する。

【0194】T4のφで、第1のアドレスの内容をGBに読み出して、ALU26に入力し、インクリメントを行う。

【0195】T4のφ#で、インクリメント結果を、WB経由で、第1のアドレスレジスタに格納する。バススイッチ34に入力した、第1のリードデータをXバスに出力すると共に、制御信号Bに含まれるSTX信号を活性状態にし、乗算器2にこの内容を入力ラッチXにラッチさせる。同時に、第2のアドレスの内容をGBに読み出して、MABに転送し、IABに出力する。

【0196】T5のφで、第2のアドレスの内容をGBに読み出して、ALU26に入力し、インクリメントを行う。T5のφ#で、インクリメント結果を、WB経由で、第2のアドレスレジスタに格納する。バススイッチ34に入力した、第2のリードデータをYバスに出力すると共に、制御信号Bに含まれるSTY信号を活性状態にし、乗算器2にこの内容を入力ラッチYにラッチさせる。MAC信号を活性状態にして、積和演算動作の開始を指示する。同時に、PC30の内容をアドレスとした命令のリードを行い、また、PC30の内容のインクリメントを行う。

【0197】T6のφで、インクリメント結果をPC30に格納する。一方、BUSY信号が活性状態になる。MAC命令では、CPU1は乗算器2とは並列に動作し、BUSY信号を無視して、次の命令の実行を開始する。

【0198】MAC命令を連続して実行した場合も、次のMAC命令がアドレス計算を行っている間に、乗算器の動作が終了するために、MAC命令実行にウェイトを挿入することはない。

【0199】プリフィックスコードを付した命令コードとすることにより、特開平-51981号公報に記載されているように、互換性を保持しつつ命令セットを拡張することができる。また、乗算器動作中に、次の積和演算を行った場合、命令フェッチとデータのアクセスを行うことができるから、命令長が長くなっても実行時間を低下させることがない。積和演算を連続的に高速に実行することができる。

【0200】乗算器2は、演算終了時点で、SYSCR13のMACSビットを参照して、オーバフローが発生していれば、MACレジスタの内容を、上限(H'7FFFFFFF)または下限(H'80000000)に固定する。

【0201】図21、22に、STMAC、LDMAC命令の動作タイミングを示す。

【0202】例えば、STMAC MACH, ER2命令などの例である。前記同様に、T2からSTMAC命令の実行が開始される。

【0203】まず、BUSY信号の状態をサンプリングする。BUSY信号が活性状態であれば、ウェイト状態になる。

【0204】T2のφ#でPCの内容がGBに読み出され、MAB33に入力されて、IABに出力される。ま

た、INC 27に入力されて、インクリメント動作が開始される。

【0205】CPU内部のクロックがロウレベルで固定され、CPUの動作を停止する。直前にMAC命令を実行した場合、BUSY信号は3ステートの期間活性状態であり、STMACH命令も3ステートウェイト状態になる。

【0206】T5でBUSY信号が非活性状態になると、T6からクロックの動作が開始される。

【0207】T6のφで、インクリメント結果がWBに出力され、PC30に格納される。STMACHまたはSTMACH信号が活性状態になって、MACレジスタの読み出しが指示される。MACレジスタの内容がXバス、Yバスに出力される。特に制限はされないものの、Xバスが上位、Yバスが下位の内容とされる。

【0208】T6のφ#で、Xバス、Yバスの内容がWBに出力されて、指定されたレジスタ(STMACH、ER2の場合は、ER2)に格納される。同時に、乗算器のフラグの内容がCCRのN、Z、Vフラグに格納される。

【0209】また、LDMAC ER1、MACL命令などの例である。

【0210】T8からLDMAC命令の実行が開始される。

【0211】T9のφで、指定されたレジスタ(LDMAC ER1、MACLの場合は、ER1)の内容が読み出される。この内容がXバス、Yバスに出力される。

【0212】T9のφ#で、LDMACHまたはLDMACH信号が活性状態になる。PC30の内容がGB経由で、MAB33とINC27に入力される。

【0213】T10のφで、インクリメントされた結果がWB経由で、PC30に格納される。また、Xバス、Yバスの内容がMACレジスタに格納される。

【0214】前記同様に、BUSY信号が活性状態の場合は、LDMAC命令も活性状態になるようにしてもよい。

【0215】CLRMAC命令は、概略LDMAC命令と同様の動作で、LDMAC命令のLDMACH、L信号と同じタイミングで、CLRMAC信号を活性状態にするようにすればよい。

【0216】図23、24に、乗算器を用いた乗算命令のタイミング図を示す。

【0217】なお、CONT22の部分に、内部のステートマシンのステップの番号を記載した。これは、基本的には、CONT22の出力のフィードバック信号で形成される。また、制御信号CPUSを用いて、乗算器を使用するか使用しないかを選択する。例えば、MULXU、WR1、ER0などのバイトサイズ・符号無し乗算の例である。前記同様に、T2から実行を開始する。

【0218】命令が解読されると、まず、T2のφ#

で、レジスタ制御信号Aによって、汎用レジスタの読みだしを指示する。読出された結果は、GB、DBおよびバススイッチ34を介して、Xバス、Yバスに出力される。

【0219】制御信号Bに含まれるSTX、STY信号に基づいて、Xバス、Yバスの内容は、T3のφで乗算器の入力ラッチにラッチされる。また、同時に、制御信号Bに含まれるMUL信号によって、乗算器に乗算を指示する。CONT22から、バイト/ワードの選択、符号付/符号無の選択、乗算/積和の選択を上記制御信号によって指示する。

【0220】乗算器は、T3のφ#で、BUSY信号を与える。また、マルチプレクサやデコーダを動作させる。T4のφで、1回目の加算を行う。T4のφで部分積を乗算結果レジスタとフィードバックラッチに格納する。これを3回繰り返す。BUSY信号が活性状態になったことに呼応して、CPUはウェイト状態になる。

【0221】T5のφでBUSYが非活性状態になって、CPUは動作を再開し、T6のφで、制御信号Cに含まれる、MULRD信号を活性状態にして、乗算結果レジスタのリードを指示する。乗算結果レジスタの内容は、Xバス、Yバスおよびバススイッチ34を経由して、T6のφ#でWBを経由して、レジスタ制御信号Cによって指定されるレジスタに格納される。同時に、乗算の結果フラグがCCR31に格納される。

【0222】前記の通り、乗算命令はBUSY信号によって、クロックが停止し、ウェイト状態となる。バイトサイズ符号無し乗算命令(MULXU、BROL、R1など)は1ウェイトが挿入され、3ステートで実行される。ワードサイズ符号無し乗算命令(MULXU、WR0、ER1など)は2ウェイトが挿入され、4ステートで実行される。なお、符号付き乗算の場合は、それぞれ、プリフィックスコードの実行が付加される。

【0223】BUSY信号によって、演算実行の終了を判定することにより、制御回路(CONT22)の論理を縮小することができる。

【0224】TESTMODE信号が活性状態になって、テストモードを指示された場合には、乗算器は1ステップの動作のみを行い、BUSY信号は非活性状態を保持する。CPUは1ステートで処理を終了する。

【0225】図25、26に、乗算器を用いない乗算命令のタイミング図を示す。乗算器を用いない乗算は、特に制限はされないものの、除算と類似のシーケンスで行うようにする。

【0226】命令が解読されると、まず、汎用レジスタの読みだしを指示する。読出された結果は、符号判定を行う。符号付/符号無の選択に対応して、符号判定を行い、除数は符号反転し、負数にする。そのほかは正数にする。

【0227】被乗数を上位、下位は0にして1ビットず

つシフトし、シフトした結果によって、下位側に乗数を加算するかを決める。その結果に対して、さらにシフトを行い、シフトした結果によって、下位側に乗数を加算を行っていく。これを8または16回繰返して、乗算結果の絶対値を得る。例えば、MULXU. B R1L, R0などのバイトサイズ・符号無し乗算の例である。前記同様に、T2から除算命令の実行が開始される。前記のような、所定の処理を行った後、T5から部分乗算を行う。

【0228】部分乗算は、左シフト処理と加算で構成される。前回の加算と今回のシフト処理を同一のALU処理で行うようにする。

【0229】T5のφ1に同期して、指定されたレジスタ（ディスティネーションレジスタRd）から被乗数を読み込み、シフト処理を行う。シフト処理の結果（部分積）がφ#に同期してWBを経由して、Rdにライトされる。また、シフトアウトされたキャリが内部で保持される。

【0230】T6のφ1に同期して、Rdから部分積を読み込み、部分乗算処理を行う。前回のキャリが”1”である場合、部分積の下位8ビットに乗数を加算し、16ビットでシフト処理を行い、最下位ビットは”0”とする。前記以外の場合、部分積に16ビットでシフト処理を行い、最下位ビットは”0”とする。かかる結果がφ#に同期してWBを経由して、Rdにライトされる。また、シフトアウトされたキャリが内部で保持される。この動作を7回繰り返す。

【0231】T13では、上記同様の判定を行い、前回のキャリが”1”である場合、部分積の下位8ビットに乗数を加算する。前記以外の場合、部分積を保持する。16ビットの積が得られる。かかる結果がφ#に同期してWBを経由して、Rdにライトされる。符号付きの場合は、T14で符号処理を行う。また、ワードサイズの場合は、部分乗算処理が8回追加される。

【0232】先に保持した符号判定結果に基づいて、積の符号処理を行う。すなわち、乗数・被乗数の一方が正数、他方が負数のときは、積の符号を反転する（0から積を引く）。

【0233】図27に、乗算命令の状態遷移図を示す。例えば、MULXU. B R1L, R0などのバイトサイズ・符号無し乗算の例である。命令の実行が開始されると、CPUS信号の状態によって分岐する。

【0234】CPUS信号が活性状態であって、乗算器の使用が許可されると、図6の動作を行う。即ち、ステップ1で、指定されたレジスタの内容を、GB、DBを経由して、X、Yバスに出力して、乗算器に供給する。BUSY信号の状態を判定する。テストモードであれば、BUSY信号は非活性状態であって、直ちにステップ2に遷移する。

【0235】BUSY信号が活性状態であると、WAI 50

T状態に遷移する。BUSY信号は非活性状態になるとステップ2に遷移する。

【0236】ステップ2では、X、Yバスの内容をWBを経由して、指定されたレジスタにライトする。例えば、乗算器のフラグの内容をCCR31に格納する。次の命令の実行を開始する。

【0237】CPUS信号が非活性状態であって、乗算器の使用が禁止されると、図25、26の動作を行う。即ち、ステップ1、2でデータアライメントなどを行った後、ステップ3から、部分乗算処理を行う。ステップ3では、GB上位に被乗数を出力し、これをシフトする。

【0238】ステップ4では、GBに部分積を、DB下位に乗数を出力し、ALU26で加算を行う。前のステップでシフトアウトしたビットが”1”であれば、加算した結果が選択され、シフトアウトしたビットが”0”であれば、GBの内容が選択され、シフトを行う。これをステップ10まで繰り返す。

【0239】ステップ11では、GBに部分積を、DB下位に乗数を出力し、ALU26で加算を行う。前のステップでシフトアウトしたビットが”1”であれば、加算した結果が選択され、シフトアウトしたビットが”0”であれば、GBの内容が選択される。シフトは行わない。

【0240】ステップ12で、命令のリードを行う。例えば、積を検査して、CCRに反映する。次の命令の実行を開始する。

【0241】図25、26に、除算命令のタイミング図を示している。例えば、DIVXU. B R1L, R0などのバイトサイズ・符号無し除算の例である。前記同様に、T2から除算命令の実行が開始される。除数の符号反転などの、所定の処理を行った後、T5から部分除算を行う。部分除算は、左シフト処理と減算で構成される。前回の減算と今回のシフト処理を同一のALU処理で行うようにする。

【0242】T5のφ1に同期して、指定されたレジスタ（ディスティネーションレジスタRd）から被除数を読み込み、シフト処理を行う。シフト処理の結果（部分剰余）がφ#に同期してWBを経由して、Rdにライトされる。また、シフトアウトされたキャリが内部で保持される。

【0243】T6のφ1に同期して、Rdから部分剰余を読み込み、部分除算処理を行う。前回のキャリが”1”である場合、または、部分剰余の上位8ビットが除数以上である場合、部分剰余の上位8ビットから除数を減算（除数の符号反転を行っている場合、除数の反転を加算）し、16ビットでシフト処理を行い、最下位ビットは”1”とする。前記以外の場合、部分剰余に16ビットでシフト処理を行い、最下位ビットは”0”とする。かかる結果がφ#に同期してWBを経由して、Rd

にライトされる。また、シフトアウトされたキャリが内部で保持される。この動作を7回繰り返す。

【0244】T13では、上記同様の判定を行い、前回のキャリが”1”である場合、または、部分剰余の上位8ビットが除数以上である場合、部分剰余の上位8ビットから除数を減算し、下位8ビットでシフト処理を行い、最下位ビットは”1”とする。前記以外の場合、部分剰余に下位8ビットでシフト処理を行い、最下位ビットは”0”とする。いずれの場合も、ビット7の値は失われる。上位8ビットに剰余、下位8ビットに商が得られる。かかる結果がφ#に同期してWBを経由して、Rdにライトされる。

【0245】符号付きの場合は、T14で符号処理を行う。また、ワードサイズの場合は、部分除算処理が8回追加される。

【0246】図28に、ALU26の概略ブロック図を示す。ALU26は、算術論理演算回路76と、選択回路77、シフト回路78、制御回路79から構成される。乗除算に直接関係のない部分は省略している。

【0247】算術論理演算回路76は、GBとDBの内容を入力して、加算、減算、論理積、論理和、排他的論理和などの演算を行い、結果を出力する。選択回路77は、算術論理演算回路の出力と、GBの内容を入力して、いずれかを選択して出力する。シフト回路78は、選択回路77の出力を入力して、シフト処理を行う。

【0248】選択回路77、シフト回路78は制御回路79によって制御される。制御回路79は、CONT22の与える制御信号と算術論理演算回路76とシフト回路78の出力によって、選択回路77の選択とシフト回路78のシフト入力を制御する。制御回路79が、前記の部分乗算、部分除算の判定を行う。条件が成立していれば、算術論理演算回路76の出力を選択し、除算の場合、1をシフト回路に入力する。条件が不成立であれば、GBの入力を選択し、除算の場合、0をシフト回路に入力する。乗算の場合のシフト回路の入力は、0とされる。

【0249】除算の部分除算と乗算の部分乗算の処理のシーケンス、及びALU26の回路構成を共通化する。除算と乗算を共通化して、CONT22の論理規模を縮小できる。

【0250】これにより、乗算器を持たないCPUを容易に提供することができる。乗算器を持つCPUにおいて不必要な乗算器を用いない乗算の論理を除算と共通化して、論理規模の増加を最低限にすることができる。

【0251】また、CPUSによって、乗算器を用いない選択を可能にすることによって、テスト性を向上することができる。テスト時に、乗算器を用いるか用いないかを選択することに両方の論理をテストの対象にすることができる。

【0252】複数命令の待避/復帰命令の命令コードは

表16の通りである。

【0253】最初に使用するレジスタ番号が、命令コード中に指定される。例えば、昭和5年3月(株)日立製作所発行『H8/500シリーズプログラミングマニュアル』に記載の複数命令の待避/復帰命令のように任意のレジスタの組合わせを指定するのではなく、連続したレジスタ番号の固定の組合わせとし、2、3、4本の固定の組合わせとしている。命令コードも、レジスタ本数に応じて3種類を用意している。

【0254】複数命令の待避命令は、待避するレジスタの本数に対応して、

STM(ERl-ERl+1), @-SP

STM(ERm-ERm+2), @-SP

STM(ERn-ERn+3), @-SP

の3種類を有する。l=0、2、4、6であり、m、n=0、4である。指定した汎用レジスタをスタックに待避する。例えば、ER0とER1をスタックに待避する場合は、

STM(ER0-ER1), @-SP

を用いる。ER0、ER1の順番でスタックにライトされ、スタックポインタ(ER7)は+8される。命令コード中のレジスタ指定部は、最初に待避されるレジスタ番号にしてある。

【0255】図29、30に複数レジスタの待避命令の実行シーケンスを示す。例えば、STM, L ER0-ER1, @-SPなどの2本の汎用レジスタを待避する例である。レジスタ指定フィールドはB'000である(B'は2進数を示す)。

【0256】前記同様に、T2から除算命令の実行が開始される。特に制限はされないものの、命令コードの第1ワードはプリフィックスコードであり、次の命令コードの動作を指定し、PCをインクリメントするほかの動作は行わない。

【0257】第2ワードの命令コードは、PUSH命令と共通にされる。

【0258】T4のφで、SPの内容をGBに読み出し、ALU26に入力する。ALU26では-4の演算を行う。なお、前記の通り、実行前のSPはスタックの先頭アドレスを示しているとする。

【0259】T4のφ#で演算結果がWBとGBに出力される。WBからSPに書き込まれ、GBからMAB33に格納される。MAB33の内容がIABに出力される。また、第1の制御信号BとRSEL2(=B'000)とによって、待避されるレジスタが選択され、レジスタ制御信号Bが生成される。

【0260】T5のφで、選択されたレジスタ(ER0)の内容がDB経由で、DBW24に転送される。

【0261】T5のφ#で、転送されたデータ(ER0の内容)の上位16ビット(Eレジスタの内容)が内部データバスに出力される。また、MAB33のインクリ

メント機能によって、IABの出力値を+2とする。

【0262】T6のφで、更に、SPの内容をGBに読み出し、ALU26に入力する。ALU26では-4の演算を行う。

【0263】T6のφ#で、DBW24に転送されたデータの下位16ビット(Rレジスタの内容)が内部データバスに出力される。ALU26の演算結果がWBとGBに出力される。WBからSPに書き込まれ、GBからMABに格納される。MAB33の内容がIABに出力される。また、第2の制御信号BによってRSELのビット0が反転される。第1の制御信号とRSEL2(=B'001)とによって、待避されるレジスタが選択され、レジスタ制御信号Bが生成される。

【0264】T7のφで、選択されたレジスタ(ER1)の内容がDB経由で、DBW24に転送される。

【0265】T7のφ#で、転送されたデータ(ER1の内容)の上位16ビット(Eレジスタの内容)が内部データバスに出力される。また、MAB33のインクリメント機能によって、IABの出力値を+2とする。

【0266】T8のφ#で、DBW24に転送されたデータの下位16ビット(Rレジスタの内容)が内部データバスに出力される。

【0267】T8のφ#以降で、前記同様に、次の次の命令の読み出しと、PC30のインクリメント(+2)を行う。

【0268】レジスタ3本を指定した場合は、実行ステート数が2ステート長くなり、SPのデクリメント(-4)と、RSELのビット1が反転される。RSELは、レジスタ指定フィールドが000の場合、010とされ、汎用レジスタER2が選択される。ライト動作が2回行われる。

【0269】レジスタ4本を指定した場合は、更に、実行ステート数が2ステート長くなり、SPのデクリメント(-4)と、RSELのビット1とビット0が反転される。RSELは、レジスタ指定フィールドが000の場合、011とされ、汎用レジスタER23が選択される。ライト動作が2回行われる。

【0270】レジスタ番号の下位ビットが固定であるので、これを命令処理の実行に従って、変更させることが容易である。例えば、2本のレジスタを待避する場合、命令コード上のレジスタ指定フィールドの下位ビットは0であるので、1回めのレジスタ指定は、レジスタ指定フィールドの値に従い、2回のレジスタ指定は、CONT22の制御に従って、レジスタ指定フィールドの下位1ビットを1に変更して、行うようにする。

【0271】一方、PUSH命令はレジスタ1本の待避であり、前記の2回目の待避動作を行わないようにされ、実行動作の共通化を図っている。

【0272】図31、32に複数レジスタの復帰命令の実行シーケンスを示す。例えば、LDM. L @ER7

+, ER0-ER1などの2本の汎用レジスタを待避する例である。レジスタ指定フィールドは001である。

【0273】図33、34に、RSEL2入力制御回路の具体的な構成、およびその動作説明を示す。この制御回路は、アンド回路75A、75B、オア回路80A、80Bから構成される。

【0274】ビット2には、オペコードのレジスタ指定フィールドのビット2がそのまま入力される。ビット1、0には、オアゲートとアンドゲートを介して入力される。オアゲートの他方の入力はSTM制御信号1、0であり、アンドゲートの他方の入力はLDM制御信号1、0の反転とされる。STM制御信号1、0およびLDM制御信号1、0は、CONT22の出力である制御信号Bに含まれる。

【0275】STM制御信号が活性状態になると、当該RSELビットは1になる。また、LDM制御信号が活性状態になると、当該RSELビットは0になる。STM、LDM命令と指定したレジスタ本数に従って、STM制御信号、LDM制御信号が生成される。

【0276】これにより、レジスタ選択回路をそのほかの命令と共通化することができる。共通化によって、物理的規模の増加を抑止できる。

【0277】図35、36に、C言語で書かれた関数と、これをCPUの命令に変換したリストの概略を示す。このリストには、オフセット(相対アドレス)、命令コード、Cラベル、Cソース及びアセンブラ命令の各項目が示されている。

【0278】C言語からCPUの命令へのコンパイルについては、例えば、平成4年9月(株)日立製作所発行『H8/300シリーズCコンパイラ』に記載されている。引数を汎用レジスタER0、ER1に設定しておくことができる。

【0279】関数Proc1では、引数をレジスタ渡しとし、これをER0に割り当てている。関数内の処理で、ER2、3、4、6を使用するため、関数処理の先頭で、

STM (ER2-ER3), @-SP

STM (ER4-ER6), @-SP

を実行して、関数の最後で、

LDM @SP+, (ER4-ER6)

LDM @SP+, (ER2-ER3)

を実行して、サブルーチンからリターン(RTS)している。

【0280】ER0、ER1は引数領域のため、関数内では使用せず、内容の待避/復帰も行わない。

【0281】また、この関数内で呼び出される関数Proc3は、引数をレジスタ渡しとし、これをER0に割り当てている。関数内の処理で、ER5を使用するため、関数処理の先頭で、1レジスタの待避
PUSH. L ER5

を実行して、関数の最後で、

POP, L ER5

を実行して、サブルーチンからリターン(RTS)している。

【0282】スタックポインタはER7と兼用であるから、ER7を待避／復帰することは意味がない。従って、タスク切替えを行う場合に使用可能なすべてのレジスタを待避する場合には、

STM @SP+, (ER0-ER3)

STM @SP+, (ER4-ER6)

の2命令を用いる。ER0からER6の順番でスタックに待避される。同様に、復帰する場合には、

LDM @SP+, (ER4-ER6)

LDM @SP+, (ER0-ER3)

の2命令を用いる。ER6からER0の順番でスタックから復帰される。

【0283】前記のように任意の組合わせを指定できないが、予め、レジスタの割当てを行っておくことにより、実質的な制約にはなりにくい。7本のレジスタを待避／復帰する場合に2命令を用いることになるが、全体的な実行ステート数やプログラム容量に対しては影響が小さい。少なくとも、1本のレジスタずつの待避／復帰命令を用いるより効果がある。後者の場合、4バイト×7、5ステート×7であるのに対して、前者では、4バイト×2、9+11ステートで実行できる。少なくとも、命令リードのためのリードサイクルや、アドレス計算のための内部動作の分を短縮して、高速化を図ることができる。

【0284】前記C言語で書かれたプログラムのように、関数乃至サブルーチンを多く用いるプログラムの高速化を実現することができる。

【0285】また、上記のような関数乃至サブルーチンの場合のほかに、割り込み処理ルーチンにおいても、同様のレジスタの待避／復帰を行う必要がある。マイクロコンピュータが機器制御などを行う場合には、割り込み処理については、割り込みのイベントが発生してから、実際の割り込み処理を実行するまでの時間を短縮することによって、リアルタイム制御性を向上することができる。複数レジスタの待避を高速に実行可能にすることにより、かかるリアルタイム制御性の向上に効果がある。

【0286】また、固定の組合わせにし、各命令の実行ステート数を固定にすることにより、内部の条件分岐を行うことをなくし、内部論理を簡潔にし、論理規模を縮小できる。マイクロプログラムによらず、ワイアードロジックなどでも容易に実現できる。マイクロプログラムによらず、ワイアードロジックなどとする事により、論理回路の高速化に寄与することができる。特に、C言語など関数乃至サブルーチンを多く用いるプログラムを高速に実行することができる。

【0287】図37、38に、割り込み例外処理のシーケ

ンスを示す。

【0288】図39に、例外処理の状態遷移図を示す。

【0289】前記同様に、T2から割り込み例外処理の実行が開始される。プリフェッチした命令はキャンセルされ、図示されない割り込み要求信号に呼応して、CONT22の入力が切り替えられる。

【0290】ステップ1の動作として、PC30のデクリメントを行う。T2のφ#で、PC30の内容を読み出して、GB経由で、INC27でデクリメント(−4)を行う。これはプリフェッチをキャンセルしたことに対応して、待避すべきPC30の値を算出する。

【0291】T3のφで、デクリメントした結果を、WB経由で一旦PC30に格納する。

【0292】ステップ2で、SPをデクリメントし、この内容をアドレスとして、PC30の内容をデータとして、ライト動作を行う。即ち、T3のφで、同時に、SPの内容を読み出して、GB経由でALU26でデクリメント(−2)を行う。

【0293】T3のφ#で、デクリメントした結果を、WB経由でSPに格納するとともに、GB経由でMAB33に転送し、IABに出力させる。

【0294】T4のφで、PC30の内容をDB経由でDBW24に転送する。DBW24の内容は、T4のφ#から、内部データバスに出力される。

【0295】ステップ3で、SPをデクリメントし、この内容をアドレスとして、PC30の上位8ビットとCCR31の内容をデータとして、ライト動作を行う。

【0296】T4のφで、同時に、SPの内容を読み出して、GB経由でALU26でデクリメント(−2)を行う。

【0297】T4のφ#で、デクリメントした結果を、WB経由でSPに格納するとともに、GB経由でMAB33に転送し、IABに出力させる。DBW24に保持したPC30の内容下位16ビットを内部データバスに出力する。

【0298】T5のφで、CCR31の内容をDB経由でDBW24に転送する。T4で格納したPC30の上位8ビットは保持される。DBW24の内容は、T5のφ#から、内部データバスに出力される。

【0299】INTM1信号が非活性状態であれば、ステップ4に遷移する。INTM1信号が活性状態であれば、ステップ12に遷移し、SPをデクリメントし、この内容をアドレスとして、EXRの内容をデータとして、ライト動作を行う。

【0300】T5のφで、SPの内容を読み出して、GB経由でALU27でデクリメント(−2)を行う。

【0301】T5のφ#で、デクリメントした結果を、WB経由でSPに格納する。

【0302】T6のφで、EXRの内容をDB経由でDBW24に転送する。T6のφ#から、内部データバス

に出力される。

【0303】ステップ4で、ベクタアドレスの内容をリードする。

【0304】T5のφ#で、同時に、VAGの内容をGB経由でMAB33に転送し、IABに出力させる。VAGには、図示されない、割り込みコントローラから与えられるベクタ番号に基づいて、ベクタアドレスを生成する。

【0305】ステップ5で、ベクタアドレスのリード動作の終了を待つ。

【0306】ステップ6で、DBR25に格納した、ベクタアドレスの内容をアドレスとして、命令のリードを行う。DBR25の内容をインクリメントし、PC30に格納する。

【0307】T8のφ#で、DBR25に格納したベクタアドレスのリード内容（分岐先の先頭アドレス）をGB経由で、MAB33に転送し、IABに出力させ、ALU26でインクリメント（+2）する。

【0308】T9のφで、インクリメントした結果を、WB経由でPC30に格納する。

【0309】ステップ7で、PC30の内容をアドレスとして、命令のリードを行い、PC30のインクリメントを行う。

【0310】T9のφ#で、PC30の内容（分岐先の先頭アドレス）をGB経由で、MAB33に転送し、IABに出力させ、ALU26でインクリメント（+2）する。リードした命令をIR21に格納する。

【0311】T10のφで、インクリメントした結果を、WB経由でPC30に格納する。

【0312】次の命令の実行を開始させる。

【0313】制御信号INTM1に従って、ステップ12を行うか、行わないかが選択され、スタックを2回行うか、3回行うかが選択される。スタックを2回行う場合には、PCとCCR31のみが待避される。SPは-4となる。上記T5の動作に相当する部分（ステップ12）が実行されない。3回行う場合には、PC30とCCR31及びEXRが待避される。SPは-6となる。

【0314】なお、ステップ1単位の動作が複数ステートにまたがっているのは、1つのCONT22の入力に対応して、複数の異なるタイミングの制御信号A、B、C及びレジスタ選択信号A、B、Cが生成されるのに対応する。

【0315】図40、41に、例外処理後のスタックの状態を示す。図41はノーマルモードを示し、図42はアドバンスモードを示している。

【0316】図42に、RTE命令の実行シーケンスを示す。

【0317】図43に、例外処理の状態遷移図を示す。

【0318】前記同様に、T2からRTE命令の実行が開始される。

【0319】ステップ1の動作として、SPの内容をアドレスとして、スタックのリードを行う。T2のφ#で、SPの内容を読み出して、GB経由で、MAB33に転送し、IABに出力させる。

【0320】T3のφで、SPの内容を読み出して、GB経由で、ALU27でインクリメント（+2）する。IABのアドレスでスタックをリードする。T3のφ#で、リードした内容をDBR25に格納する。

【0321】INTM1信号が非活性状態であれば、ステップ2に遷移する。INTM1信号が活性状態であれば、ステップ10に遷移し、リードした結果をEXRに格納する。SPをインクリメントし、この内容でリードを行う。

【0322】T3のφ#で、同時に、インクリメントした結果を、WB経由でSPに格納する。また、GBを経由して、MAB33に転送し、IABに出力させる。

【0323】T4のφで、DBR25の内容をGBに読み出し、これをALU27に inputsする。T4のφ#で、ALU27はGBから入力した内容をそのまま、WBに出力し、EXRに格納する。

【0324】ステップ2で、リードした結果をCCR31に格納する。MAB33に格納した内容を、MAB33でインクリメントさせる。この内容でリードを行う。なお、MAB33のインクリメント機能は、特開平4-333153号公報などに記載されている。

【0325】T5のφで、DBR25の内容をGBに読み出し、これをALU27に inputsする。T5のφ#で、ALU27はGBから入力した内容をそのまま、WBに出力し、CCR31に格納する。

【0326】ステップ3で、SPの内容をインクリメント（+4）する。

【0327】T6のφで、SPの内容を読み出して、GB経由で、ALU26でインクリメント（+4）する。

【0328】T6のφ#で、インクリメントした結果を、WB経由でSPに格納する。

【0329】ステップ4で、DBR25に格納した、スタックから復帰したPC30の内容をアドレスとして、命令のリードを行う。DBR25の内容をインクリメントし、PC30に格納する。

【0330】T6のφ#で、同時に、DBR25に格納したベクタアドレスのリード内容（分岐先の先頭アドレス）をGB経由で、MABに転送し、IABに出力させ、ALUでインクリメント（+2）する。

【0331】T7のφで、インクリメントした結果を、WB経由でPC30に格納する。

【0332】ステップ5で、PC30の内容をアドレスとして、命令のリードを行い、PC30のインクリメントを行う。

【0333】T7のφ#で、PC30の内容（分岐先の先頭アドレス）をGB経由で、MAB33に転送し、I

ABに出力させ、ALU26でインクリメント(+2)する。リードした命令をIRに格納する。

【0334】T8のφで、インクリメントした結果を、WB経由でPC30に格納する。次の命令の実行を開始させる。

【0335】例えば、INTM1信号が0レベルの場合には、前記従来CPU（例えば、前記平成5年6月（株）日立製作所発行『H8/300Hシリーズプログラミングマニュアル』に記載のCPU）と同一のスタックの構造とされる。命令コードが共通であることと相俟って、従来CPUによって書かれたプログラムをそのまま実行することができる。

【0336】新たな、コンディションコードや割込みマスクビットやトレースビットなどを追加する場合には、これに対応したプログラムを作成することになるから、スタックの構造が異なっても実質的な問題はない。割込みマスクビットを追加するなどして、使い勝手を向上することができる。

【0337】なお、前記の通りINTM1ビットがSYSCRに存在し、このビットの状態がINTM1信号に反映されるようになっている。リセット後に、かかるSYSCRの設定を行うことにより、EXRを使用するかしないかが選択される。

【0338】図44にCONT22の一部の論理を示

表21

ビット7：シングルステップイネーブル（SSTPE）

シングルステップを設定します。

ビット7	説明
SSTPE	
0	シングルステップ解除（初期値）
1	シングルステップ設定

【0343】

【表22】

表22

ビット6：RTB割込イネーブル（RTBINTE）

RTB命令実効終了後の割込受付の許可／禁止を選択します。

ビット6	説明
RTBINTE	
0	RTB命令実効終了後の割込を禁止（必ずI命令実行します）
1	RTB命令実効終了後の割込を許可（初期値）

【0344】

【表23】

す。このCONT22は、アンド回路86A～86Dによって構成される。

【0339】EXRを使用しない、すなわち、INTM1ビットを”0”にクリアすると、EXRのビットは全て”0”とみなされ、設定値は、無視されるようにされる。

【0340】次に、図4におけるエミュレーション用インタフェース39に含まれる制御レジスタ41の構成を示す。この制御レジスタ41は、以下説明するように、

(1) ASEコントロールレジスタ D（ASECRD）、(2) ブレークコントロールレジスタ A B（BRCRA、B）、(3) ブレークアドレスレジスタ A、B（BARA、B）、(4) ブレークアドレスマスクレジスタ A、B（BAMRA、B）、および(5) ASE専用スタックレジスタ（BRKSTKR）から構成されている。

【0341】図45に、(1) ASEコントロールレジスタ D（ASECRD）の構成を示す。このレジスタは8ビットリード／ライト可能なレジスタで、シングルステップの設定、RTB命令実行後の割込制御、多重ブレークの許可禁止、ウインドウ機能を指定する。各ビットの内容を表21乃至表24に示す。

【0342】

【表21】

表 2 3

ビット5：多重ブレークイネーブル (MBIE)

ブレークモード中のブレーク割込 (ASEBRK#端子によるブレーク割込) の許可/禁止を選択します。

ビット5	説明
MBIE	
0	ブレークモード中のブレーク割込を禁止 (初期値)
1	ブレークモード中のブレーク割込を許可

10

【0345】

【表24】

表 2 4

ビット1、0：ウィンドウセレクト1、0 (WD1、WD0)

ブレークモード時のウィンドウ機能を設定します。これによって、ブレークモード時のユーザ空間へのCPUのデータアクセスを許可することができます。

ビット1	ビット0	説明
WD1	WD0	
0	0	ウィンドウ機能を解除 (初期値)
0	1	CPUデータリードサイクルに対し、ウィンドウを設定
1	0	CPUデータライトサイクルに対し、ウィンドウを設定
1	1	CPUデータリード/ライトサイクルに対し、ウィンドウを設定

【0346】図46に、(2) ブレークコントロールレジスタ A B (BRCRA、B) の構成を示す。このレジスタは、(a) BRCRA、(b) BRCRBからなり、各々は8ビットのリード/ライトが可能なレジス

タで、それぞれPCブレークのチャンネルA、Bの制御を行う。各ビットの内容を表25乃至表28に示す。

【0347】

【表25】

表 2 5

ビット7：コンディションマッチフラグA/B (CMFA/B)

チャンネルA/Bの設定した条件が成立したことを示します。

ビット7	説明
CMFA/B	
0	[クリア条件] 0をライトしたとき (初期値)
1	[セット条件] チャンネルA/Bの設定した条件が成立したとき

【0348】

【表26】

表 2 6

ビット5：トリガアウトプットイネーブルA/B (TOEA/B)

チャンネルA/Bのトリガ出力信号 (ASEBKTOA/B#) の出力の許可/禁止を選択します。

ビット6	説明
TOEA/B	
0	トリガ出力 (ASEBKTOA/B#) は禁止 (ハイレベル固定) (初期値)
1	トリガ出力 (ASEBKTOA/B#) は許可 (条件成立時1ステートロウレベル出力)

【0349】

【表27】

表 27

ビット2、1：コンディションセレクトA/B1、0 (CSELA/B1、CSELA/B0)

チャンネルA/Bのブレーク条件を選択します。指定した条件が成立し、アドレス比較が一致すると、PCブレークが要求されます。対象はCPUのみです。

ビット2	ビット1	説明
CSELA/B1	CSELA/B0	
0	0	命令実行 (初期値)
0	1	CPUデータリードサイクル
1	0	CPUデータライトサイクル
1	1	CPUデータリード/ライトサイクル

【0350】

【表28】

表 28

ビット0：ブレーク割込イネーブルA/B (BIEA/B)

チャンネルA/BのPCブレークの許可/禁止を選択します。

ビット0	説明
BIEA/B	
0	チャンネルA/BのPCブレーク禁止 (初期値)
1	チャンネルA/BのPCブレーク許可

【0351】図47に、(3) ブレークアドレスレジスタ A、B (BARA、B) の構成を示す。このレジスタは、(a) BARA、(b) BARBからなり、各々は32ビットのリード/ライトが可能なレジスタで、それぞれPCブレークのチャンネルA、Bのアドレスを指定する。32ビットのレジスタをバイトサイズに分割して、BARR、E、H、Lと表記される場合もある。最上位のBARRはリザーブされている。リードすると不定値が読み出される。ライトは無効である。

【0352】図48に、(4) ブレークアドレスマスクレジスタ A、B (BAMRA、B) の構成を示す。このレジスタは、(a) BAMRA、(b) BAMRBからなり、各々は32ビットのリード/ライトが可能なレジスタで、それぞれPCブレークのチャンネルA、Bのアドレス比較のマスクを行うビットを指定する。BAMRのビットを"1"にセットすると、このビットに対応するアドレスのビットは、アドレス比較対象から除外され

る。32ビットのレジスタをバイトサイズに分割して、BAMRR、E、H、Lと表記される場合もある。最上位のBARRはリザーブされている。リードすると不定値が読み出される。ライトは無効である。

【0353】図49に、(5) ASE専用スタックレジスタ (BRKSTKR) の構成を示す。このレジスタは、6バイト (48ビット) のリード/ライトが可能なレジスタで、ユーザモード⇄ブレークモードの遷移時に、スタック領域として使用する。ユーザのSPは使用せず、保持される。スタックされるリソースおよびスタックの構造は、MCU動作モード (ノーマルモード/アドバンスモード) および、制御レジスタの設定 (SYSCRのINTM1ビット) によって相違される。表29にこのレジスタの使用方法を示す。

【0354】

【表29】

表29

ASE専用スタックレジスタの使用方法

相対アドレス	ノーマルモード		アドバンスモード	
	INTM1=0	INTM1=1	INTM1=0	INTM1=1
0		EXR		EXR
1		リザーブ		リザーブ
2	CCR	CCR	CCR	CCR
3	CCR	CCR	PC	PC
4	PC	PC		
5				

【0355】エミュレーション用ソフトウェアの実行状態への遷移（ブレーク）時には、固定アドレスのブレークスタックレジスタを使用するようにする。ブレーク例外処理や、ブレークからのリターン命令時には、ユーザのスタックポインタ（ER7）を使用せず、固定的なスタックアドレスを生成する。かかるスタックアドレスの生成はEMLSP29による。

【0356】図50に、EMLSP29の構成を示す。このEMLSP29は、クロックバッファで構成される。

【0357】かかるクロックバッファの内、ビット23～10は1固定、ビット5、4、0は0固定、ビット9～6は、外部からの指定を入力する。また、ビット3、2、1はCONT22の制御信号を入力する。CMOS回路で構成する場合、必要に応じて論理反転を用いればよい。クロックバッファの出力はGBに接続されている。また制御信号mは、CONTの制御信号とクロック（φ#）の論理積信号である。

【0358】通常のレジスタ回路が、データを保持するためのラッチ回路を持たなければならないが、EMLSP29は、これを持たず、小型化を図っている。

【0359】従って、ブレークスタックレジスタの先頭アドレスは、B'000000であって、64kバイト単位で16通りのアドレスを選択可能とされる。マイクロコンピュータの内部I/Oレジスタの配置によって、アドレスを変更できる。

【0360】ブレーク例外処理の実行シーケンスは、図37、38と同様であり、そこでのSP（ER7）の読み出しに代わって、EMLSP29を読み出すようにする。

【0361】この場合、最初（T3のφ）は下位アドレスをB'000110として、デクリメントした内容のB'000100がスタックのアドレスとされる。

【0362】2回目（T4のφ）は下位アドレスをB'000100として、デクリメントした内容のB'000010がスタックのアドレスとされる。

【0363】3回目（T5のφ）は下位アドレスをB'

000010として、デクリメントした内容のB'000000がスタックのアドレスとされる。3回目は、INTM1信号が活性状態のときに有効である。読み出されるビット2、1は、CONT22の制御信号によって選択する。

【0364】リターン命令の実行シーケンスは、図42と同様であり、SP（ER7）の読み出しに代わって、EMLSP29を読み出すようにする。

【0365】最初（T2のφ#）はINTM1信号によって異なり、INTM1信号が非活性状態であれば、下位アドレスをB'000010として、INTM1信号が活性状態であれば、下位アドレスをB'000000として、読み出す。これらがスタックのアドレスとなる。

【0366】2回目（T3のφ）は、INTM1信号が活性状態である場合に有効であり、下位アドレスをB'000010として、デクリメントした結果をアドレスとしてリードを行う。

【0367】3回目はMAB33のインクリメントによってアドレスを生成し、EMLSP29は使用しない。

【0368】これにより、固定的な出力回路として論理規模を縮小できる。ユーザに公開されない資源による論理規模の増大を最小限にすることができる。

【0369】図51に、ブレーク例外処理の実行タイミングを示す。実行シーケンスは図37、38の例外処理タイミングと同様である。

【0370】前記同様に、T2から割り込み例外処理の実行が開始される。プリフェッチした命令はキャンセルされ、図示されないブレーク要求信号に呼応して、CONT22の入力が切り替えられる。

【0371】T3のバス動作が行われない期間に、ブレークモードを示す信号BRKAK#が活性状態になる。

【0372】図52に、ブレーク制御論理の回路構成を示す。この回路は、アンド回路81A乃至81G、オア回路82A乃至82D、フリップフロップ83から構成されている。

【0373】CPUに対するブレーク要求は、3要因が

存在する。第1はBRK端子による要求である。第2はアドレス比較Aによる要求であり、これは、BRKCRのBIEAビットによって許可される。第3はアドレス比較Bによる要求であり、これは、BRKCRのBIEBビットによって許可される。なお、かかるアドレス比較は、前記の通り、 $(CA23 \cdot AR23 + \neg CA23 \cdot \neg AR23 + AMR23) \cdot \dots \cdot (CAN \cdot ARn + \neg CAN \cdot \neg ARn + AMRn) \cdot \dots \cdot (CA0 \cdot AR0 + \neg CA0 \cdot \neg AR0 + AMR0)$ と表現される。

(\neg は論理反転を示す)。

【0374】これらの論理和信号が、ブレイク要求として、CPUに与えられる。

【0375】ブレイクモードでは、MBIEビットの状態によってBRK端子によるブレイク要求の許可禁止が選択される。即ち、ブレイクモードでMBIEビットが“0”にクリアされている場合は、ブレイク要求が抑止される。アドレス比較によるブレイク要求は、ブレイクモードで禁止される。

【0376】また、MBIEビットは、フリップフロップで構成され、BRKAK信号の反転信号で“0”にクリアされる。かかるフリップフロップの入力は、所定のデータバスのビットであって、クロックは、ブレイクモード信号とアドレスデコード信号とライト信号の論理積信号とされる。かかるアドレスデコード信号は、CPUの出力するアドレスがBRKCRの存在するアドレスになったとき、活性状態とされる。即ち、ブレイクモードでのみライト可能とされる。

【0377】即ち、ブレイクモードに遷移した直後は、ブレイク要求が禁止状態であって、不所望のブレイクの多重例外処理（スタックした内容の破壊）が禁止される。また、BRKCRのSSTPビットと、BRKAK信号の反転信号との論理積が、シングルステップブレイク要求として、CPUに与えられる。

【0378】シングルステップブレイク要求と、RTB命令実行信号との論理積信号と、ブレイク要求が、CPU内部で、CPUブレイク例外処理要求として認識される。これらの例外処理の内容は共通とされる。

【0379】CPUはRTB命令実行時には、かかるシングルステップブレイク要求を無視する。

【0380】上記実施例によれば、以下の作用効果を得るものである。

【0381】(1) 既存の命令セットと互換性を維持しつつ、乗算器を内蔵することに当っては、ポストインクリメントレジスタ間接のアドレッシングモードのみをサポートすることによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。また、アドレスレジスタの補正をレジスタ間演算命令で行い、これを1ステートで実行することができ、アドレッシングの柔軟性を向上することができる。さらに、積和演算をCPUの内部

動作（ポストインクリメントのアドレス計算）と並行に行うことによって、実行ステート数の短縮を行うことができる。

【0382】(2) 乗算器を利用して乗算命令を実行することにあたっては、乗算の結果（積、フラグ）を直接汎用レジスタ、CCRに格納するようにして、直ちに結果を利用できるようにし、実質的な乗算の実行速度を向上することができる。また、積和演算の結果（MAC）をリード（STMAC）すると同時に、乗算器内部で保持したフラグをCCRに格納することによって、積和演算結果の利用や判定を容易に行うことができ、使い勝手を向上することができる。

【0383】(3) 乗算器を取外し可能にすることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位CPUを実現し、論理的・物理的規模を縮小し、製造費用を低減した別のマイクロコンピュータを容易に開発することができる。また、汎用的な乗算命令を、乗算器によらずにサポートすることによって、かかる別のマイクロコンピュータにおける使い勝手の低下を防止できる。さらに、乗算器によらない乗算命令を除算と同一のシーケンスで実行するようにして、乗算器を持つマイクロコンピュータにおいても冗長な論理を最低限にすることができる。

【0384】また、乗算器使用するか使用しないかの制御信号を与えて制御することによって、テスト性を向上したり、エミュレータを共通化したりすることができる。全体的な開発効率を向上することができる。さらにまた、乗算器を削除し、小型化したCPUを用いて、マイクロコンピュータを構成することによって、半導体集積回路の論理規模・物理的規模を縮小して、製造費用の縮小を図ることができる。

【0385】(4) 乗算器とCPUを一体に構成して、乗算器・CPU間の配線を短縮して、物理的規模を縮小する。また、高速化に寄与することができる。

【0386】(5) 乗算器のテストモードを設定して、このときの乗算器の処理を1ステップのみにすることによって、論理規模の増加を最低限にして、テストの容易性を向上することができる。テストステップを短縮することができる。

【0387】(6) 内部動作のパイプラインに対応して、入出力タイミングの異なるレジスタ選択回路を複数持つことにより、実質的に1命令/1ステート実行を行うことができる。

【0388】(7) 複数レジスタの退避/復帰命令を持ち、この組み合わせを固定的にすることによって、論理規模の縮小を図ることができる。レジスタの本数の異なる命令を複数命令サポートすることによって、使い勝手の低下を防ぐことができる。また、複数レジスタの退避/復帰命令を関数（サブルーチン）の入力口/出力口で実行することによって、C言語などで記述された場合のよう

に、関数の使用頻度が高い場合に、処理速度を特に向上することができる。さらに、割り込み例外処理ルーチンの先頭で複数レジスタの退避を用いることにより、リアルタイム性の向上を図ることができる。

【0389】(8) EXRの有効/無効を切り換えることで、互換性を維持することと、機能拡張とを両立することができる。EXRを無効とし、例外処理において退避/復帰を行わないようにすることで、スタックの節約と、割り込み応答時間の高速化に寄与することができる。また、互換性を維持する。さらに、EXRを有効とする

ことで、割り込みマスケレベルを拡張したり、トレース機能を追加したりして、使い勝手を向上することができる。

【0390】(9) エミュレータ用の固定的なスタックポインタを持つことによって、ユーザプログラムとエミュレーションプログラムの遷移時に、ユーザーのスタックポインタとは独立して、固定的なアドレスに対して退避および復帰が行われるから、エミュレータのソフトウェア、ハードウェアの開発を容易にすることができる。

【0391】また、エミュレータ用のスタックポインタを固定的にすることによって、ユーザに公開しない資源を最小限の論理的・物理的規模にすることができる。ユーザプログラムからエミュレーションプログラムへの遷移(ブレーク)を多重に行うことを禁止することを可能にすることによって、不所望のスタックの内容の破壊を防止することができる。

【0392】以上本発明者等によってなされた発明を実施例に限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能である。実施例を相互に組み合わせ使用することもできる。

【0393】例えば、CPUの命令セットやレジスタ構成は変更可能である。内部バス幅なども変更可能である。但し、命令の大部分の命令コード長より、小さいバス幅でないことが望ましい。

【0394】また、乗算器の内部構成なども種々変更可能である。16ビット×6ビットを3回繰り返すのではなく、16ビット×4ビットを4下位繰り返すようにしてもよい。命令実行ステートと論理的規模に鑑みて選択すればよい。

【0395】さらに、飽和演算の指定、乗算器あり/なしの指定、テストモードの指定方法が種々変更可能であることは言うまでもない。

【0396】さらにまた、乗算器に限らず除算器を内蔵するものであっても良い。

【0397】また、互換性を維持すべき対象は、前記例に限定されない。一般的に、CPUの例外処理時に、コントロールレジスタの内容を待避することは行われており、そのほかのCPUについても、本発明を適用して、互換性を維持しつつ、コントロールレジスタの機能を拡張することができる。

【0398】さらに、シングルチップマイクロコンピュータのその他の機能ブロックについても何等制約されない。

【0399】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるシングルチップマイクロコンピュータに適用した場合について説明したが、それに限定されるものではなく、その他のデータ処理装置にも適用可能であり、本発明は少なくとも、複数の動作モードを選択して動作するデータ処理装置に適用することができる。

【0400】

【発明の効果】本題において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【0401】(1) 乗算器をCPUに内蔵することによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。乗算器による乗算の結果を汎用レジスタCCRに反映させ、かかる結果を直ちに利用可能にして、処理速度を高速にすることができる。

【0402】(2) 乗算器を取外し可能に(独立して)設けることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位CPUを実現し、論理的・物理的規模を縮小し、製造費用の低減に寄与することができる。

【0403】(3) 複数レジスタの退避/復帰命令を持ち、この組み合わせを固定的にすることによって、論理規模の縮小を図ることができ、また、高速化を図ることができる。

【0404】(4) コントロールレジスタの有効/無効を切り換えることで、スタックの節約と、割り込み応答時間の高速化に寄与できるとともに、互換性を維持することができる。

【0405】(5) エミュレータ専用の固定スタックポインタを持つことにより、エミュレータをサポートすることができる。また、論理規模の増加を最低限にして、エミュレータの設計を容易にすることができ、さらにエミュレータ専用スタックポインタの一部のアドレスを、CPU外部から与えるようにして、スタックレジスタをリロケート可能にし、マイクロコンピュータのアドレス配置などに容易に対応することができる。

【図面の簡単な説明】

【図1】本発明の実施例によるマイクロコンピュータの構成を示すブロック図である。

【図2】本実施例のマイクロコンピュータのシステムコントロールレジスタの構成図である

【図3】本実施例のマイクロコンピュータに用いられるCPUの命令フォーマットの構成図である。

【図4】本実施例のマイクロコンピュータに用いられるCPUと乗算器を示すブロック図である。

【図5】本実施例のマイクロコンピュータにおける制御レジスタの1ビットの構成図である。

【図6】本実施例のマイクロコンピュータにおける制御信号の設定方法の一例を示すブロック図である。

【図7】本実施例のマイクロコンピュータにおける制御信号の設定方法の一例を示す概略図である。

【図8】本実施例のマイクロコンピュータに用いられるCPUのレジスタの構成図である。

【図9】本実施例のマイクロコンピュータに用いられるCPUのレジスタの構成図である。

【図10】本実施例のマイクロコンピュータに用いられるCPUのレジスタの使用方法を説明するブロック図である。

【図11】本実施例のマイクロコンピュータに用いられるCPUのレジスタのスタックの状態の説明図である。

【図12】本実施例のマイクロコンピュータに用いられるCPUの基本動作のタイミング図である。

【図13】本実施例のマイクロコンピュータに用いられるCPUの基本動作のタイミング図である。

【図14】本実施例のマイクロコンピュータに用いられる乗算器の構成を示すブロック図である。

【図15】本実施例のマイクロコンピュータに用いられる乗算器による演算方法の説明図である。

【図16】本実施例のマイクロコンピュータに用いられる乗算器のフラグの実現方法の説明図である。

【図17】本実施例のマイクロコンピュータに用いられる乗算器のフラグの実現方法の説明図である。

【図18】本実施例のマイクロコンピュータに用いられるバススイッチの構成を示すブロック図である。

【図19】本実施例のマイクロコンピュータにおけるM 30
AC命令の動作のタイミング図である。

【図20】図19に連続する動作のタイミング図である。

【図21】本実施例のマイクロコンピュータにおけるSTMAC命令およびLDMAC命令の動作のタイミング図である。

【図22】図21に連続する動作のタイミング図である。

【図23】本実施例のマイクロコンピュータにおいて乗算器を用いた場合の乗算命令の動作のタイミング図である。

【図24】図19に連続する動作のタイミング図である。

【図25】本実施例のマイクロコンピュータにおいて乗算器を用いない場合の乗算命令の動作のタイミング図である。

【図26】図25に連続する動作のタイミング図である。

【図27】本実施例のマイクロコンピュータにおける乗算命令の状態遷移図である。

【図28】本実施例のマイクロコンピュータに用いられる演算器の構成を示すブロック図である。

【図29】本実施例のマイクロコンピュータにおける複数レジスタの退避命令の実行シーケンス図である。

【図30】図29に連続する動作のタイミング図である。

【図31】本実施例のマイクロコンピュータにおける複数レジスタの復帰命令の実行シーケンス図である。

【図32】図31に連続する動作のタイミング図である。

【図33】本実施例のマイクロコンピュータにおけるRSEL 2入力制御回路の構成を示すブロック図である。

【図34】図33の動作の説明図である。

【図35】本実施例のマイクロコンピュータに適用されるC言語による変換リストの概略例である。

【図36】本実施例のマイクロコンピュータに適用されるC言語による変換リストの概略例である。

【図37】本実施例のマイクロコンピュータにおける割込例外処理の実行シーケンス図である。

【図38】図37に連続する動作のタイミング図である。

【図39】本実施例のマイクロコンピュータにおける例外処理の状態遷移図である。

【図40】本実施例のマイクロコンピュータにおける例外処理後のスタックの状態の説明図である。

【図41】本実施例のマイクロコンピュータにおける例外処理後のスタックの状態の説明図である。

【図42】本実施例のマイクロコンピュータにおけるRTE命令の実行シーケンス図である。

【図43】本実施例のマイクロコンピュータにおける例外処理の状態遷移図である。

【図44】本実施例のマイクロコンピュータにおける制御回路の構成の説明図である。

【図45】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。

【図46】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。

【図47】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。

【図48】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。

【図49】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。

【図50】本実施例のマイクロコンピュータにおけるエミュレーションスタックポインタの構成の説明図である。

【図51】本実施例のマイクロコンピュータにおけるブレーク割込みシーケンスの実行タイミング図である。

【図52】本実施例のマイクロコンピュータにおけるブレーク制御処理の回路構成図である。

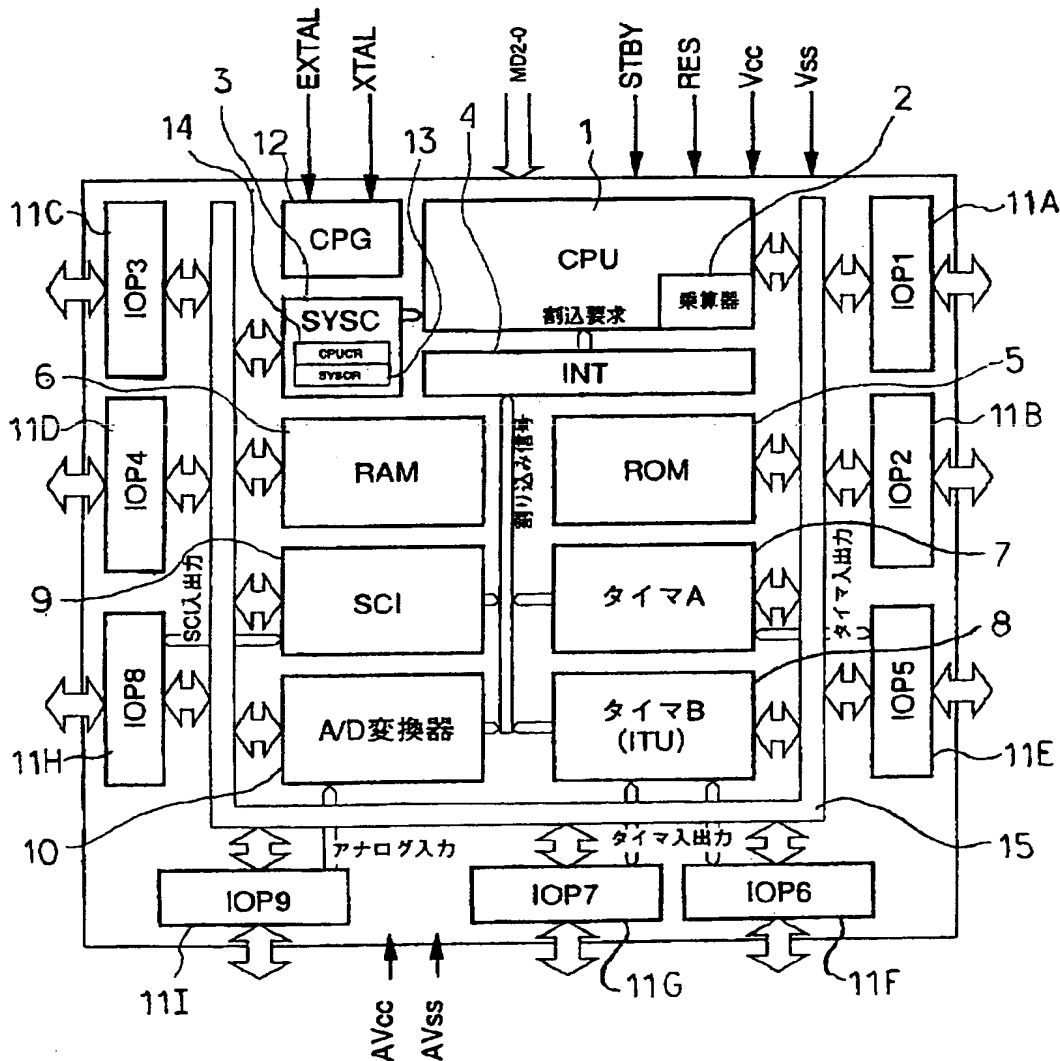
【符号の説明】

1…CPU、2…乗算器、3…システムコントローラ (SYSC)、4…割込コントローラ (INT)、5…ROM、6…RAM、9…シリアルコミュニケーションインターフェース (SCI)、13…システムコントロールレジスタ (SYSCR)、14…制御レジスタ (C PUCR)、21…命令レジスタ (IR)、22…命令デコーダ・制御回路 (CONT)、23…レジスタセクタ (RSEL)、24…ライトデータバッファ (DBW)、25…リードデータバッファ (DBR)、26、27…演算器、29…エミュレータスタックポインタ (EMLSP)、30…プログラムカウンタ (PC)、31…コンディションレジスタ (CCR)、32…拡張

レジスタ (EXR)、33…アドレスバッファ (MAB)、34…バススイッチ、38…エミュレーション用プロセッサ、39…エミュレーション用インターフェース、44…インターフェースケーブル、48…エミュレーションメモリ、49…ブレーク制御回路、50…リアルタイムトレース回路、58…CMOSインバータ回路、65A～65C…デコーダ、66A～66C、71A、71B、77…選択回路、67…加算器、72A～72C…拡張回路、76…算術論理演算回路、78…シフト回路、79…制御回路、75A、75B、81A～81G、86A～86D…アンド回路、80A、80B、82A～82D…オア回路、83…フリップフロップ。

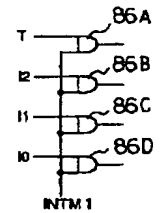
【図1】

図 1



【図44】

図 4 4



【図 2】

図 2

ビット:	7	6	5	4	3	2	1	0
	MACS	-	INTM1	INTM0	NMIEG	-	-	RAME
初期値:	0	0	0	0	0	0	0	1
R/W:	R/W	-	R/W	R/W	R/W	-	-	R/W

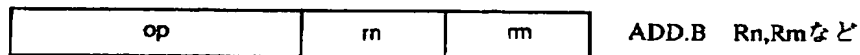
【図 3】

図 3

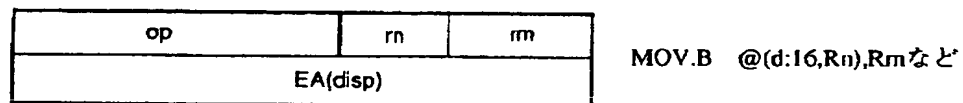
- (1) オペレーションフィールドのみ



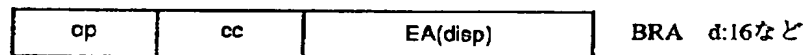
- (2) オペレーションフィールドとレジスタフィールド



- (3) オペレーションフィールド、レジスタフィールドおよびEA拡張部

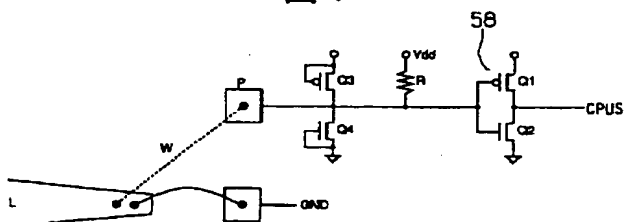


- (4) オペレーションフィールド、EA拡張部およびコンディションフィールド



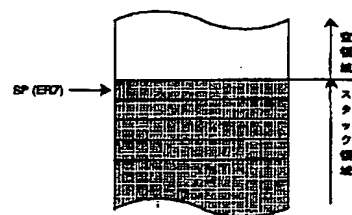
【図 7】

図 7



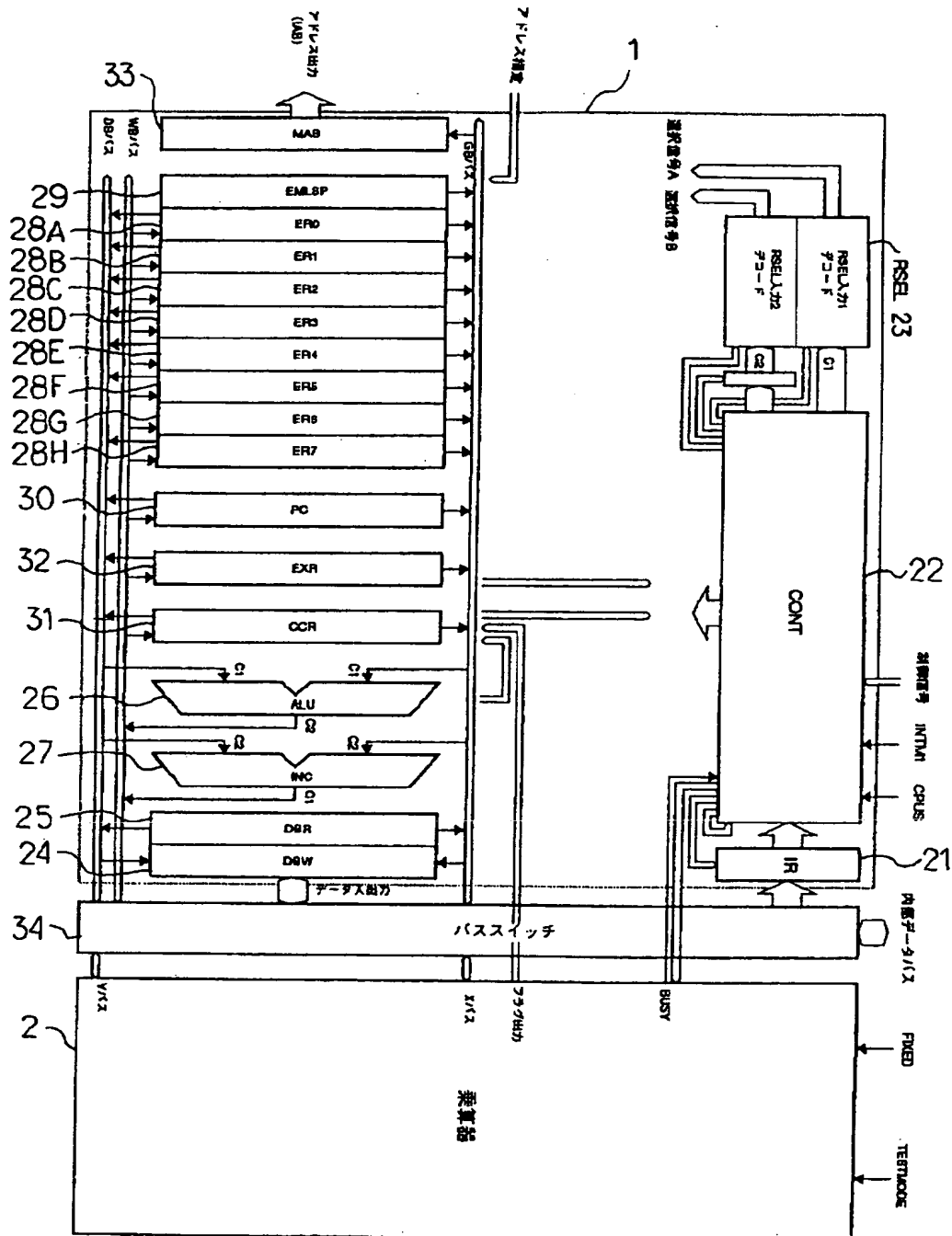
【図 11】

図 11



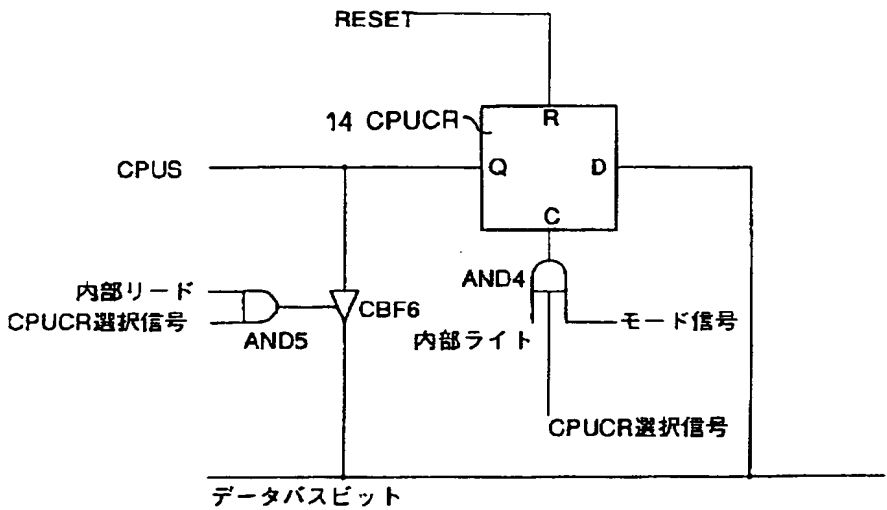
【図4】

図 4



【図 5】

図 5



【図 8】

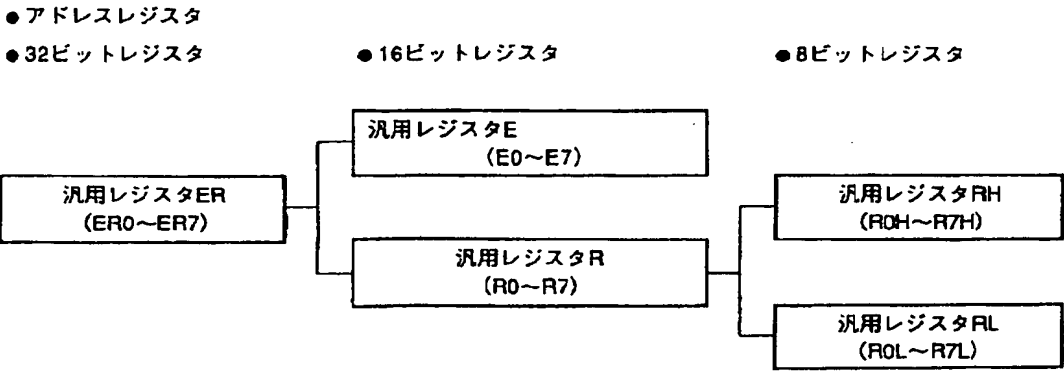
図 8

汎用レジスタ (Rn)

	15	0 7	0 7	0
ER0	E0		R0H	R0L
ER1	E1		R1H	R1L
ER2	E2		R2H	R2L
ER3	E3		R3H	R3L
ER4	E4		R4H	R4L
ER5	E5		R5H	R5L
ER6	E6		R6H	R6L
ER7(SP)	E7		R7H	R7L

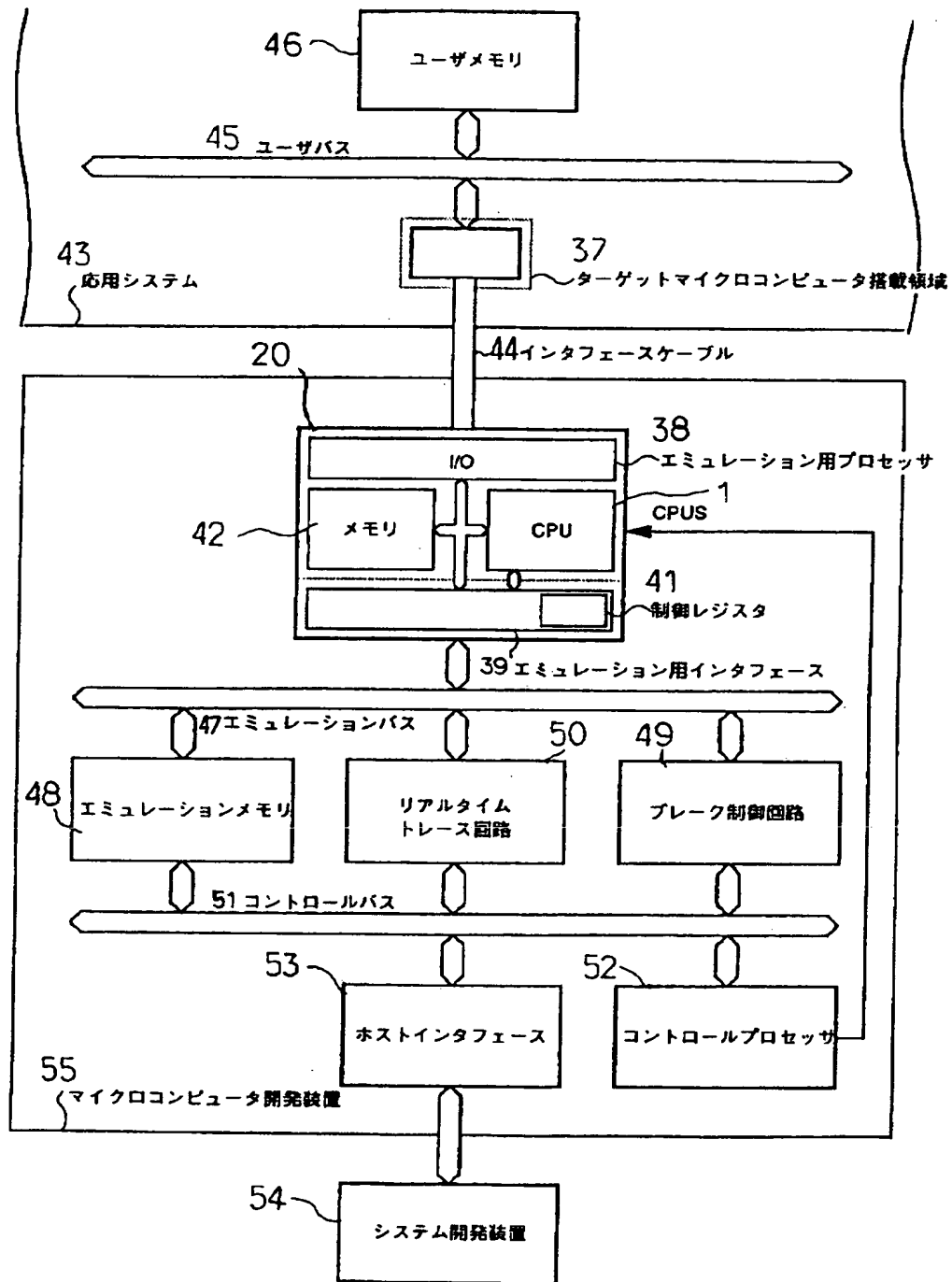
【図 10】

図 10



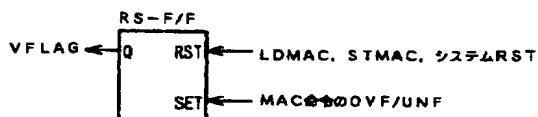
【図6】

図 6



【图 16】

图 16



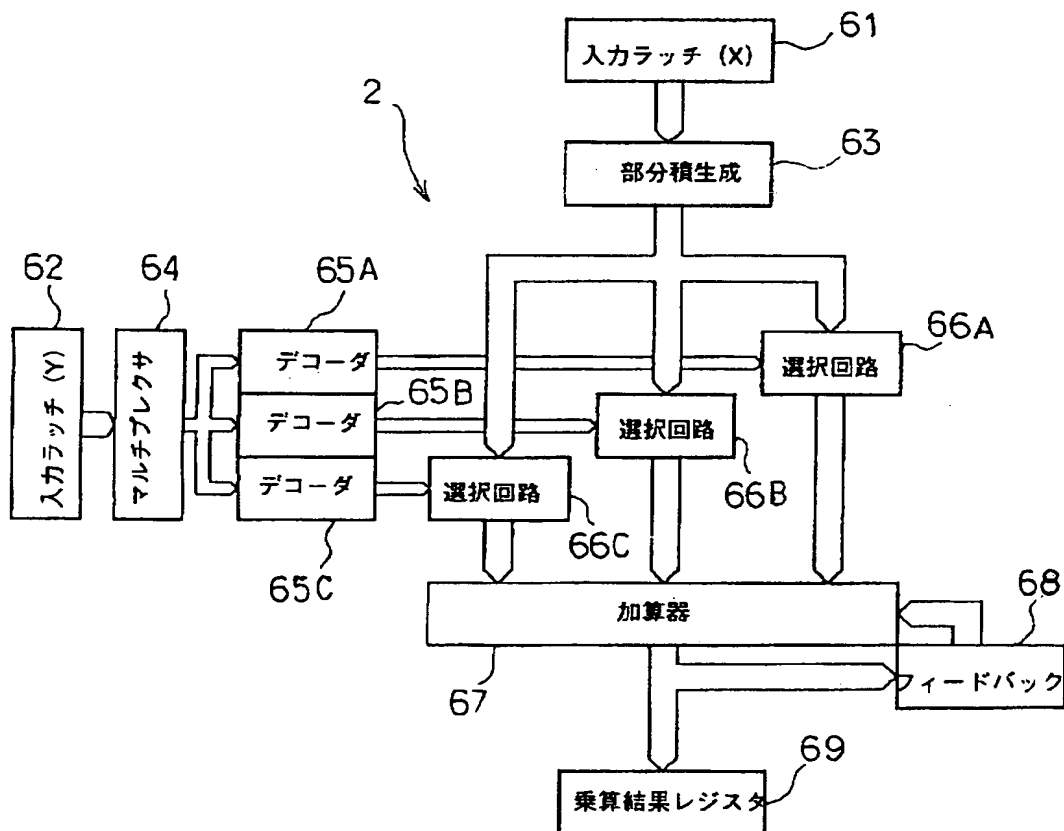
【図 4 5】

图 45

ビット:	7	6	5	4	3	2	1	0
	SSTPTE	RTBINTP	MBIE	-	-	-	WDI	WDO
初期値:	不定/0	不定/1	不定/0	不定	不定	不定	0/0	0/0
R/W:	(R/W)	(R/W)	(R/W)	-	-	-	(R/W)	(R/W)

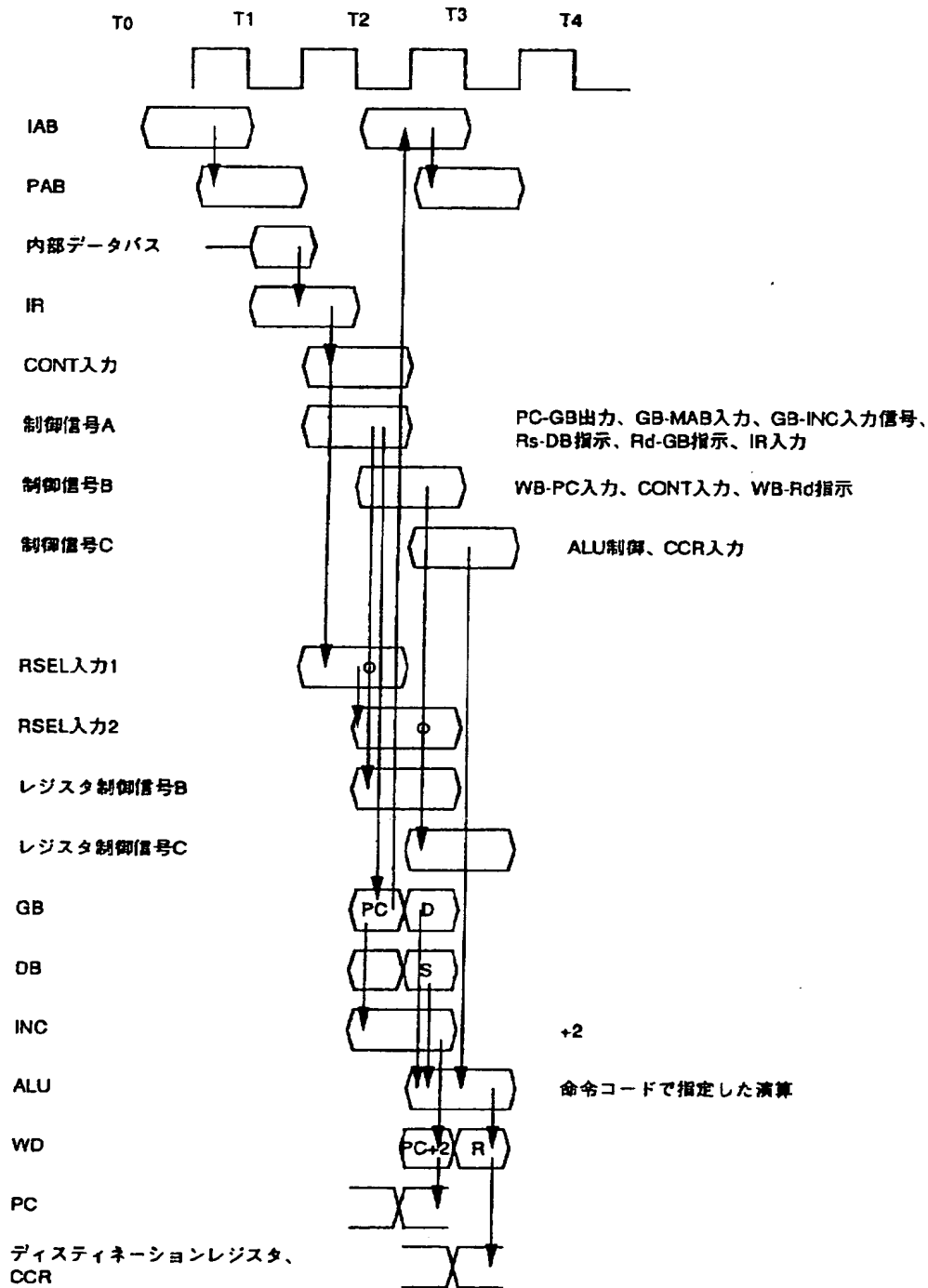
【図 14】

图 14



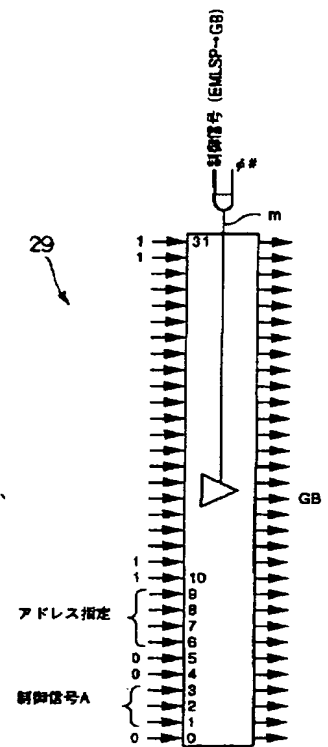
【図12】

図12



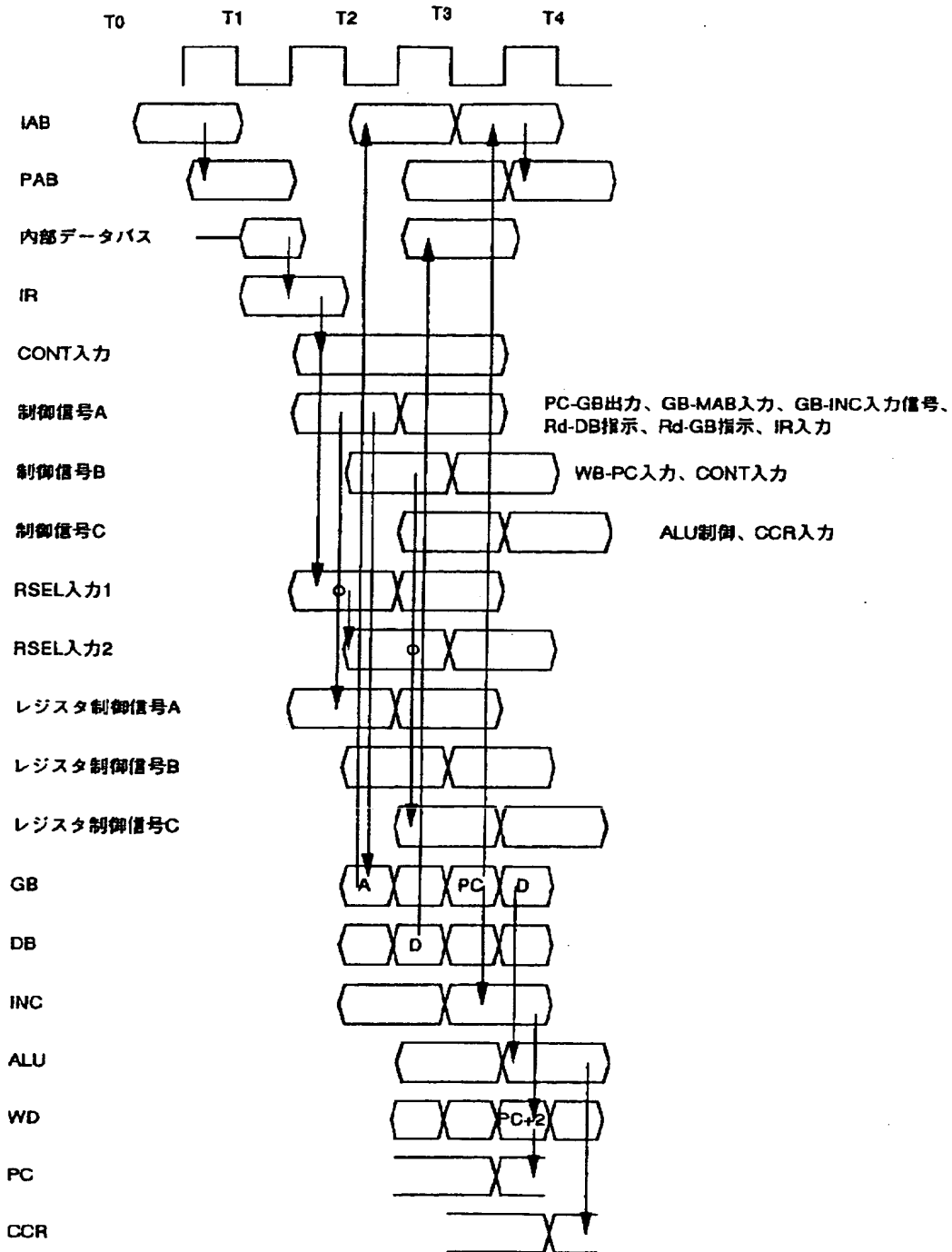
【図50】

図50

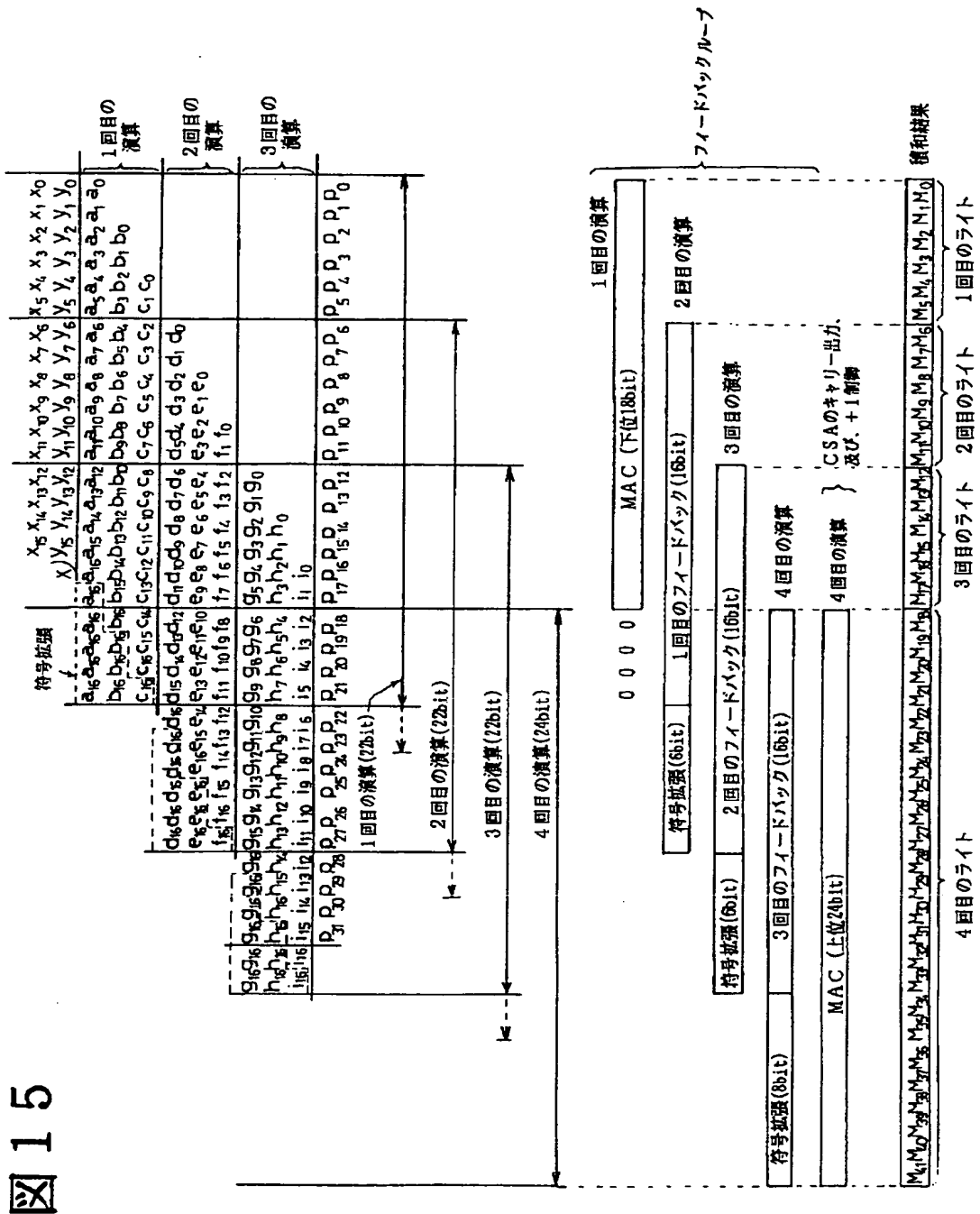


【図 1 3】

図 1 3



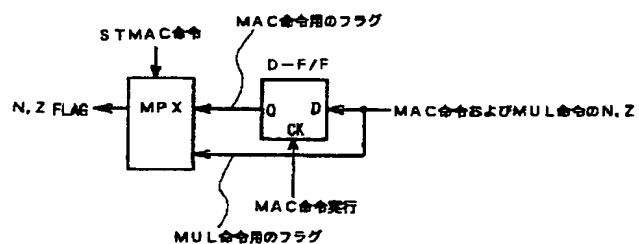
【図15】



【图 17】

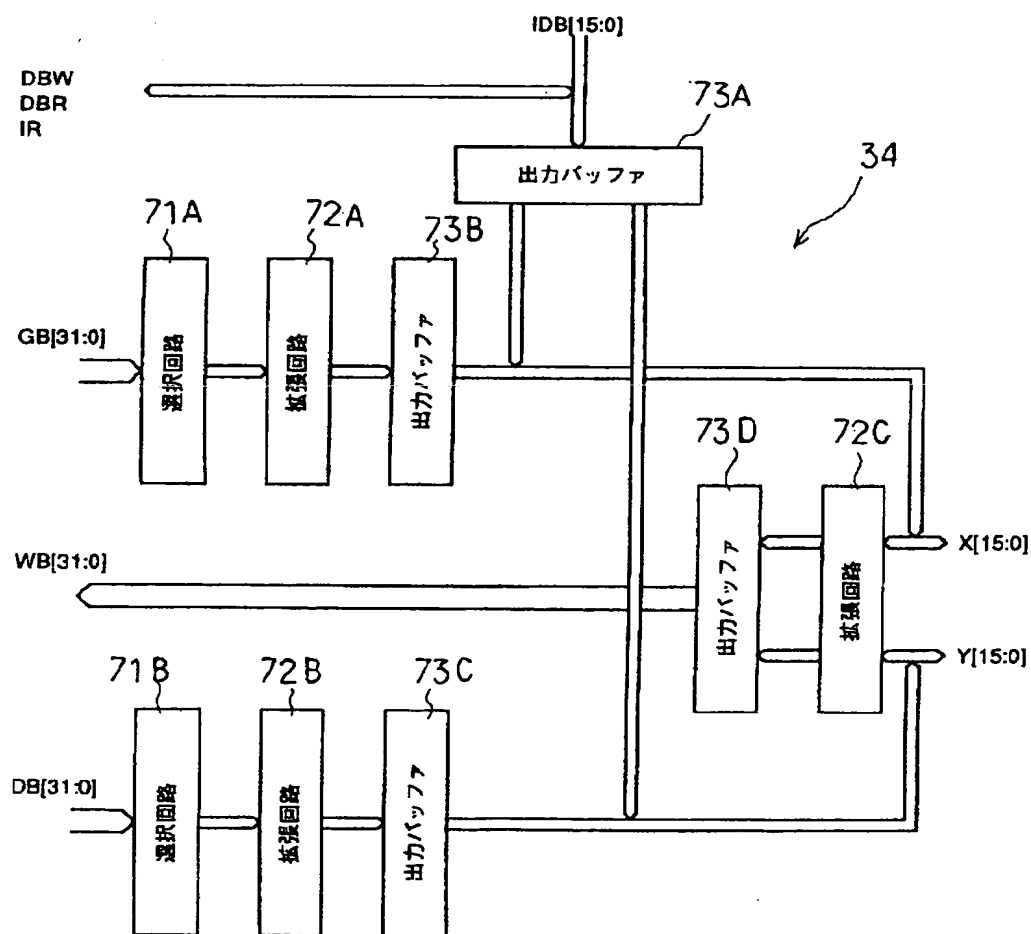
图 17

Nフラグ、Zフラグ



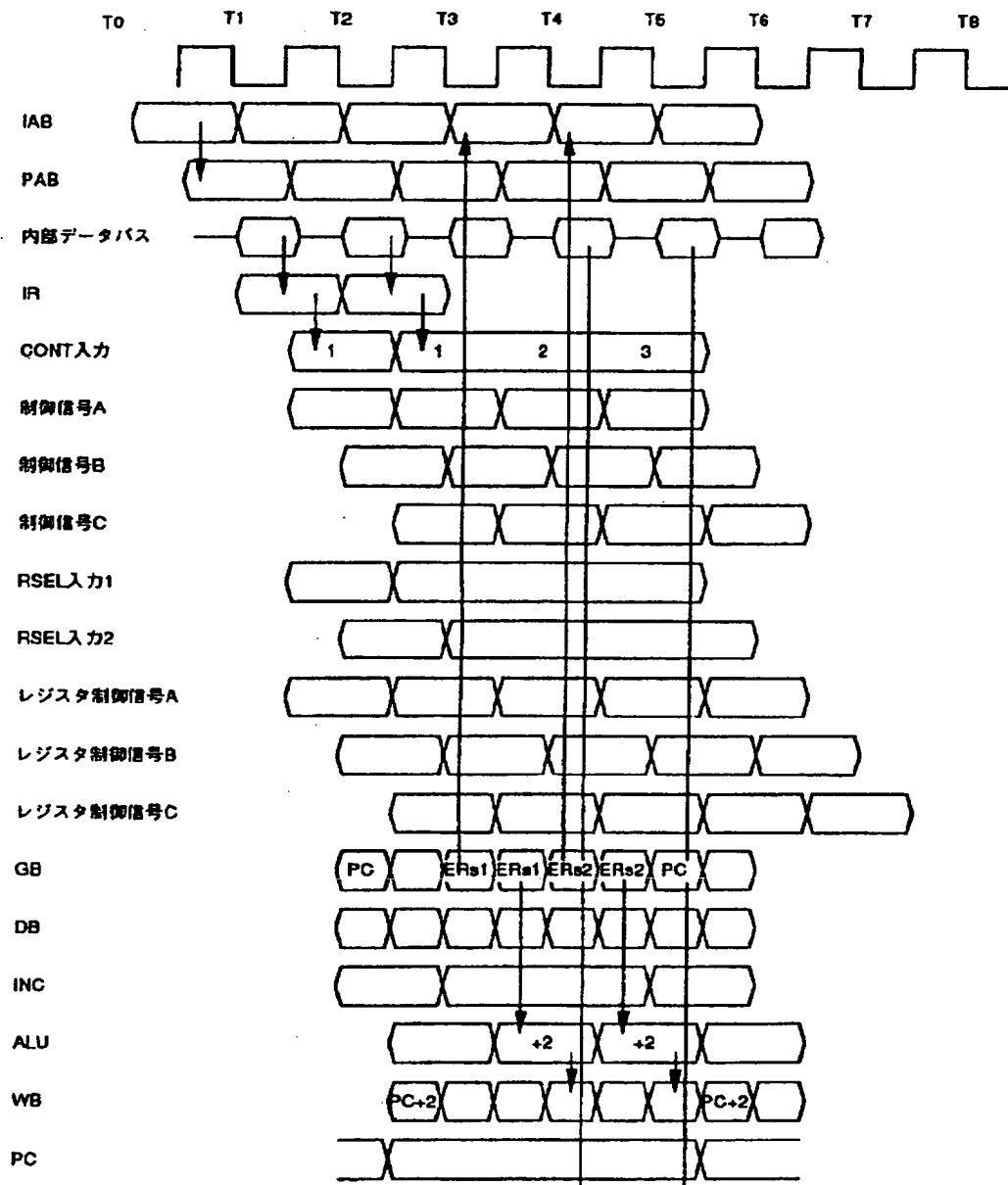
【图 18】

图 18



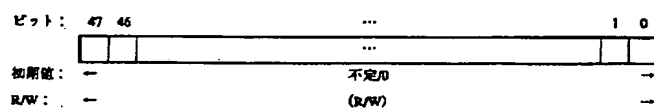
【図19】

図 19



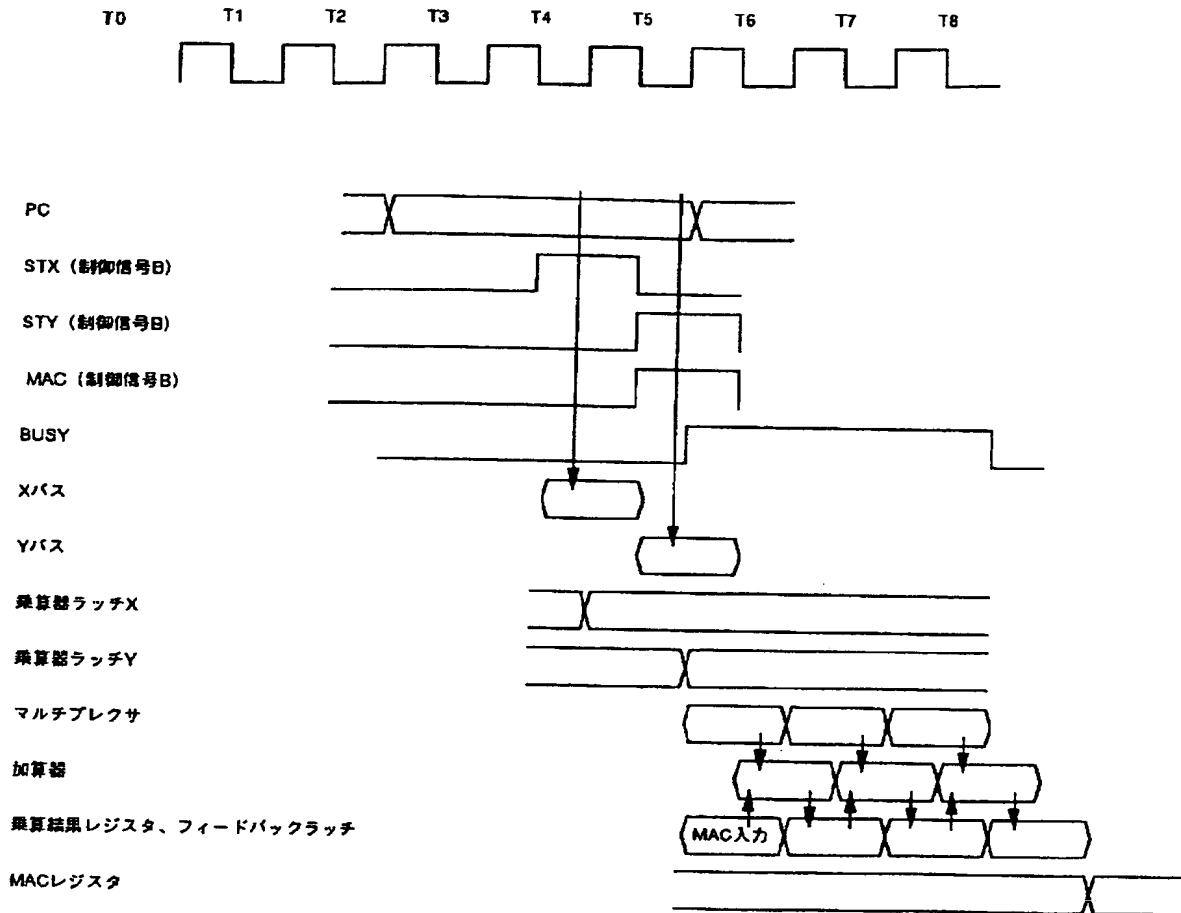
【図49】

図 49



【図20】

図 20



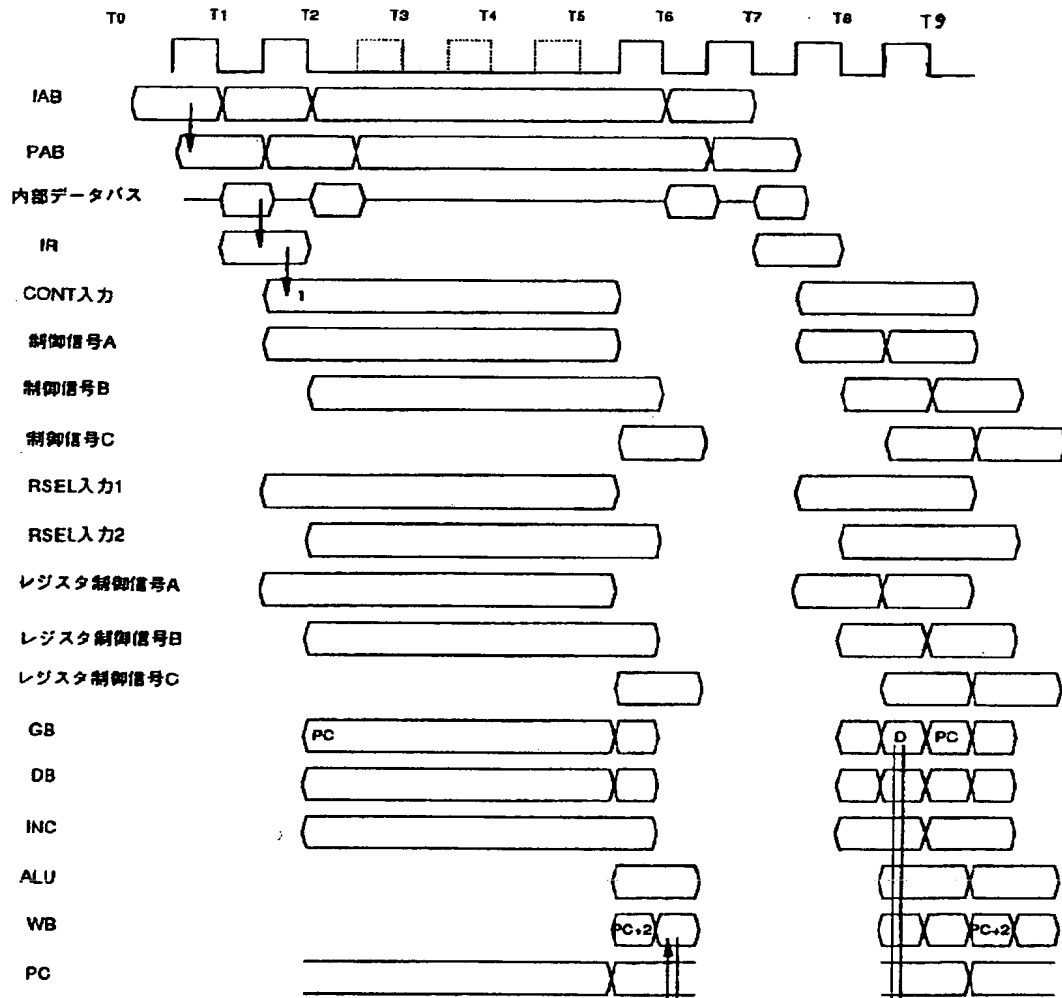
【図34】

図 3 4

命令	RSEL2 入力初期値	2 回目動作	3 回目動作	4 回目動作
2 本STM	**0	STM制御信号0=1		
3 本STM	*00	STM制御信号0=1	STM制御信号1=1	
4 本STM	*00	STM制御信号0=1	STM制御信号1=1	STM制御信号1、0=1
2 本LDM	**1	LDM制御信号0=1		
3 本LDM	*10	LDM制御信号1=1 STM制御信号1=1	LDM制御信号1=1	
4 本LDM	*11	LDM制御信号0=1	LDM制御信号1=1	LDM制御信号1、0=1

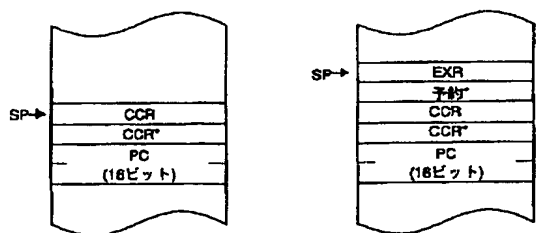
【図 2 1】

図 2 1



【図 4 0】

図 4 0



(a) 割込み制御モード0、1

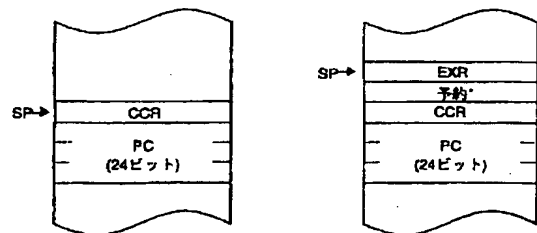
(b) 割込み制御モード2、3

* リターン時には無視されます。

例外処理終了後のスタックの状態 (ノーマルモード)

【図 4 1】

図 4 1



(a) 割込み制御モード0、1

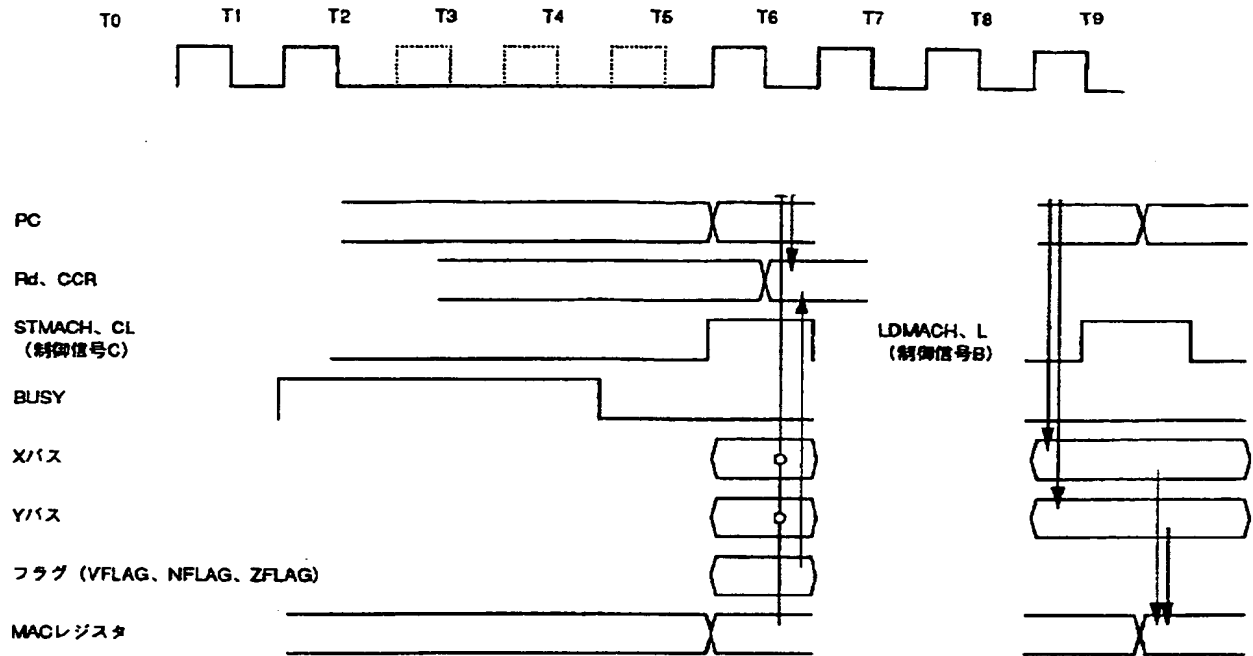
(b) 割込み制御モード2、3

* リターン時には無視されます。

例外処理終了後のスタックの状態 (アドバンスモード)

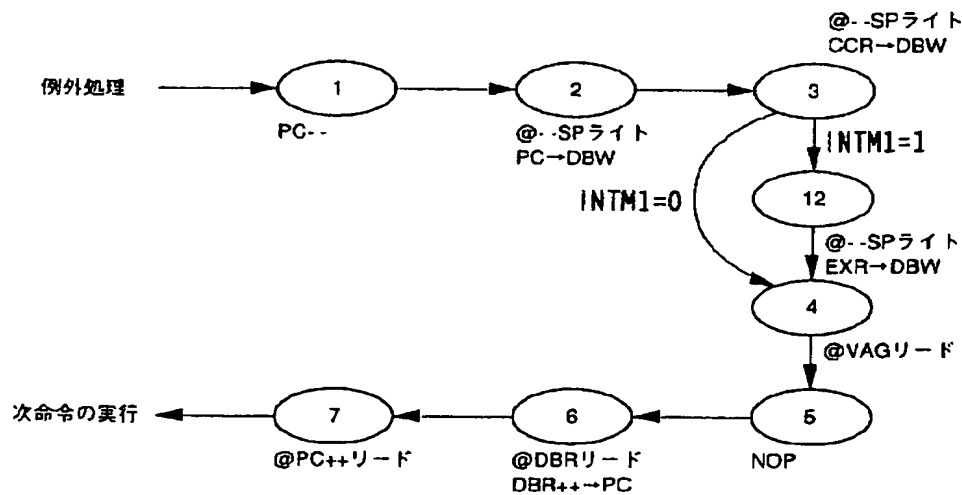
【図22】

図 2 2



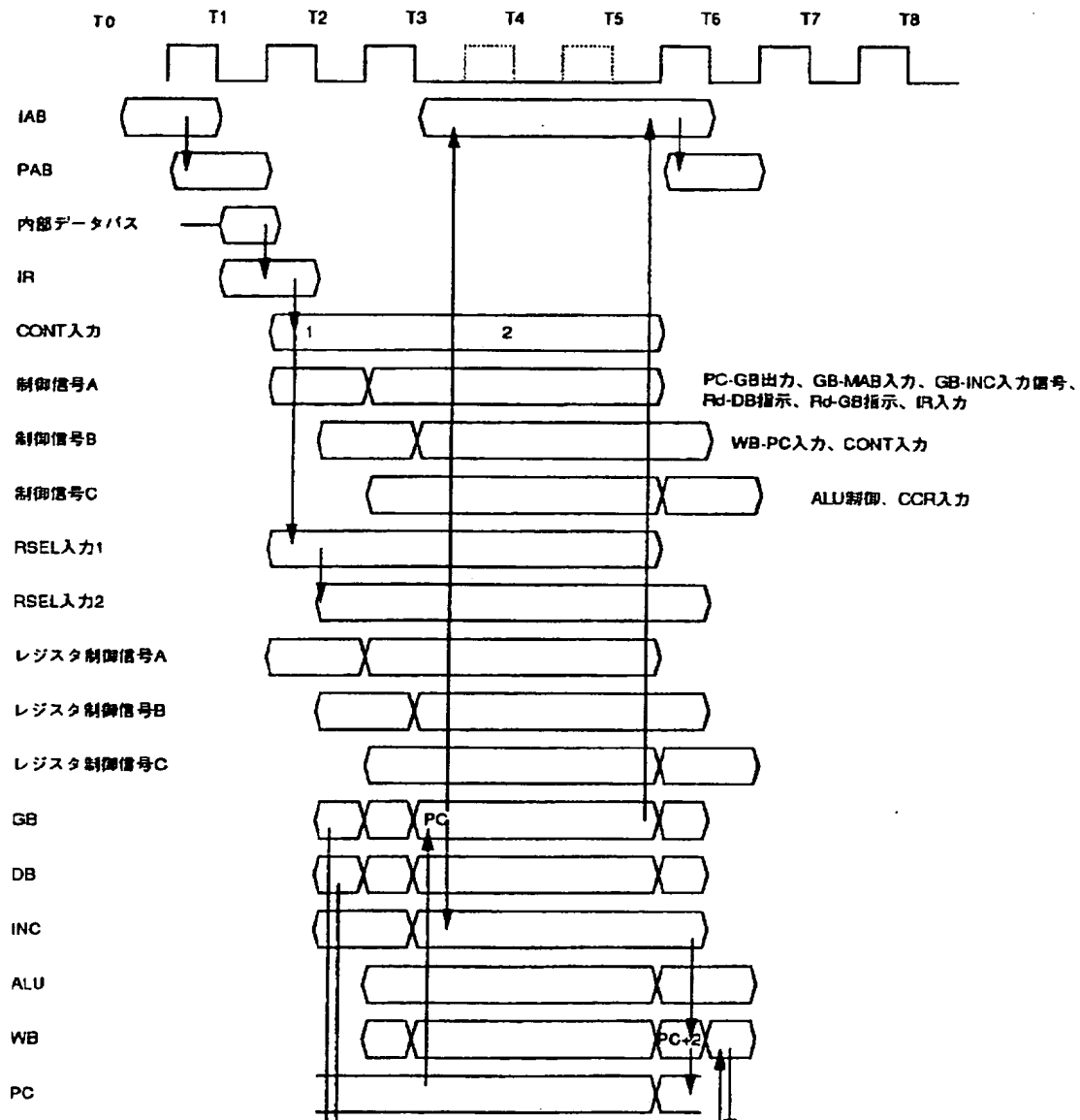
【図39】

図 3 9



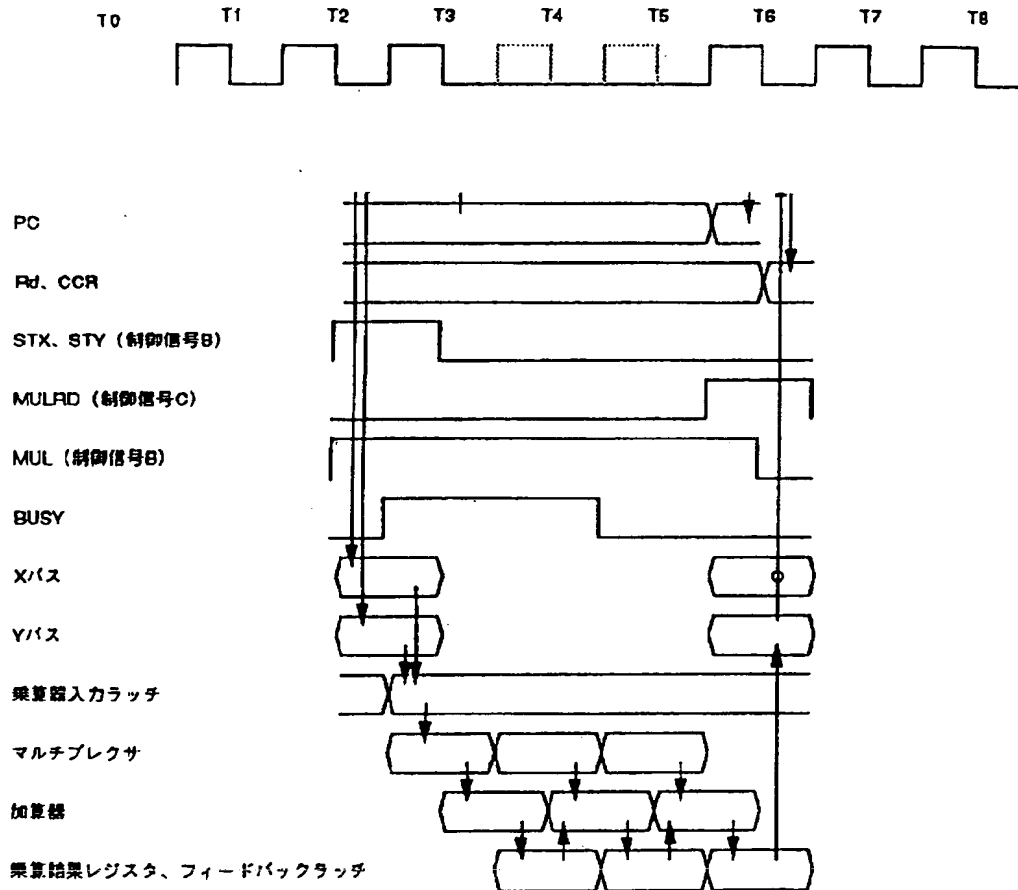
【図23】

図 2 3



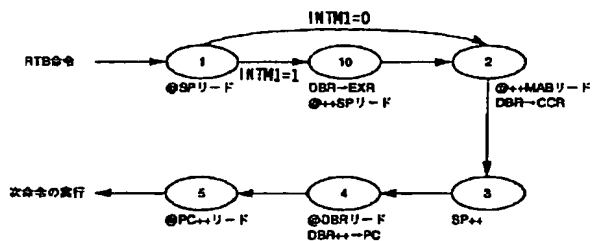
【図 2 4】

図 2 4



【図 4 3】

図 4 3



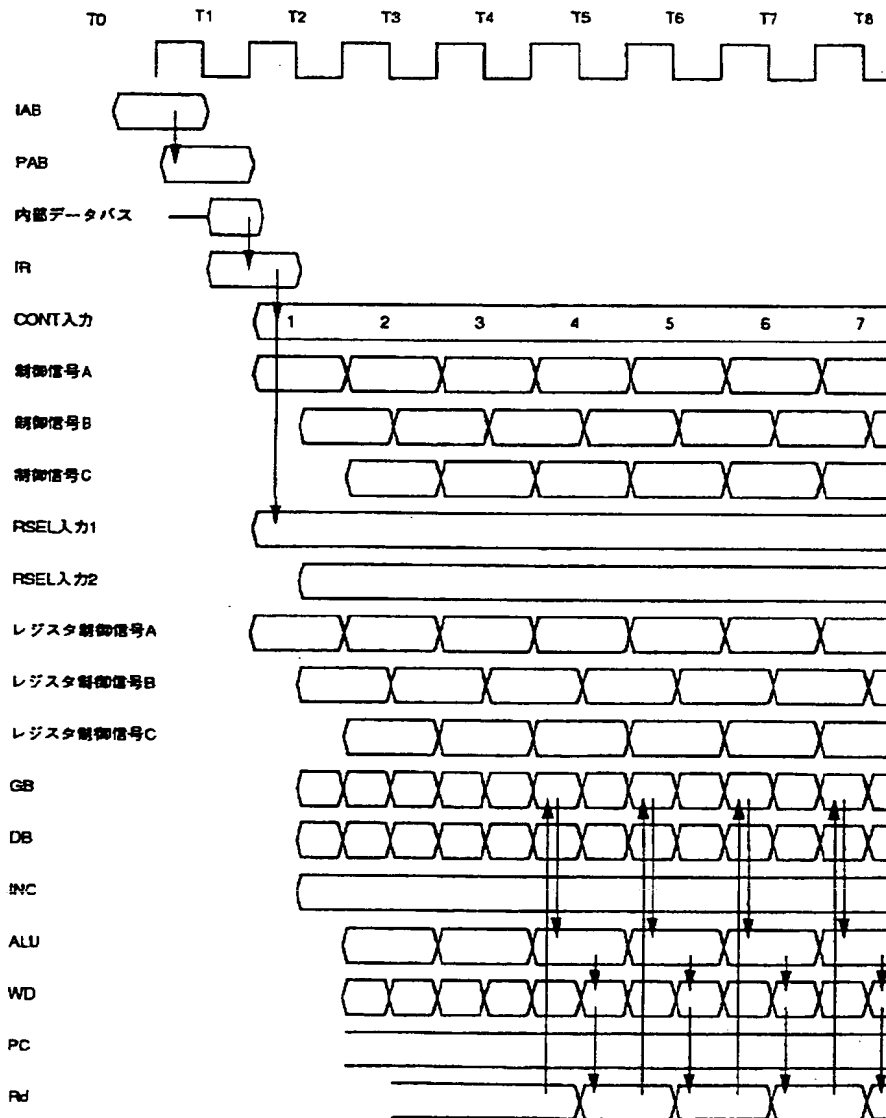
【図 4 6】

図 4 6

(a)BRCRA								
ビット:	7	6	5	4	3	2	1	0
CMFA	-	-	-	-	-	-	-	-
初期値:	不定/0	不定	不定/0	不定	不定	不定/0	不定/0	不定/0
R/W:	(R/W)	-	(R/W)	-	-	(R/W)	(R/W)	(R/W)
(b)BRCRB								
ビット:	7	6	5	4	3	2	1	0
CMFB	-	-	-	-	-	-	-	-
初期値:	不定/0	不定	不定/0	不定	不定	不定/0	不定/0	不定/0
R/W:	(R/W)	-	(R/W)	-	-	(R/W)	(R/W)	(R/W)

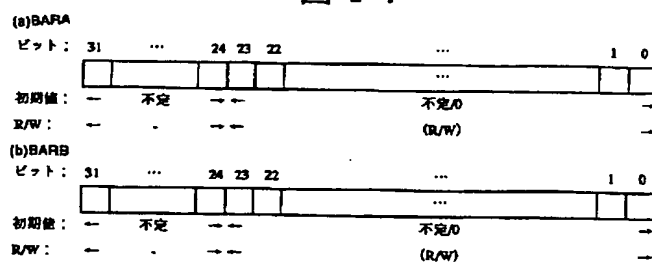
【図 25】

図 25



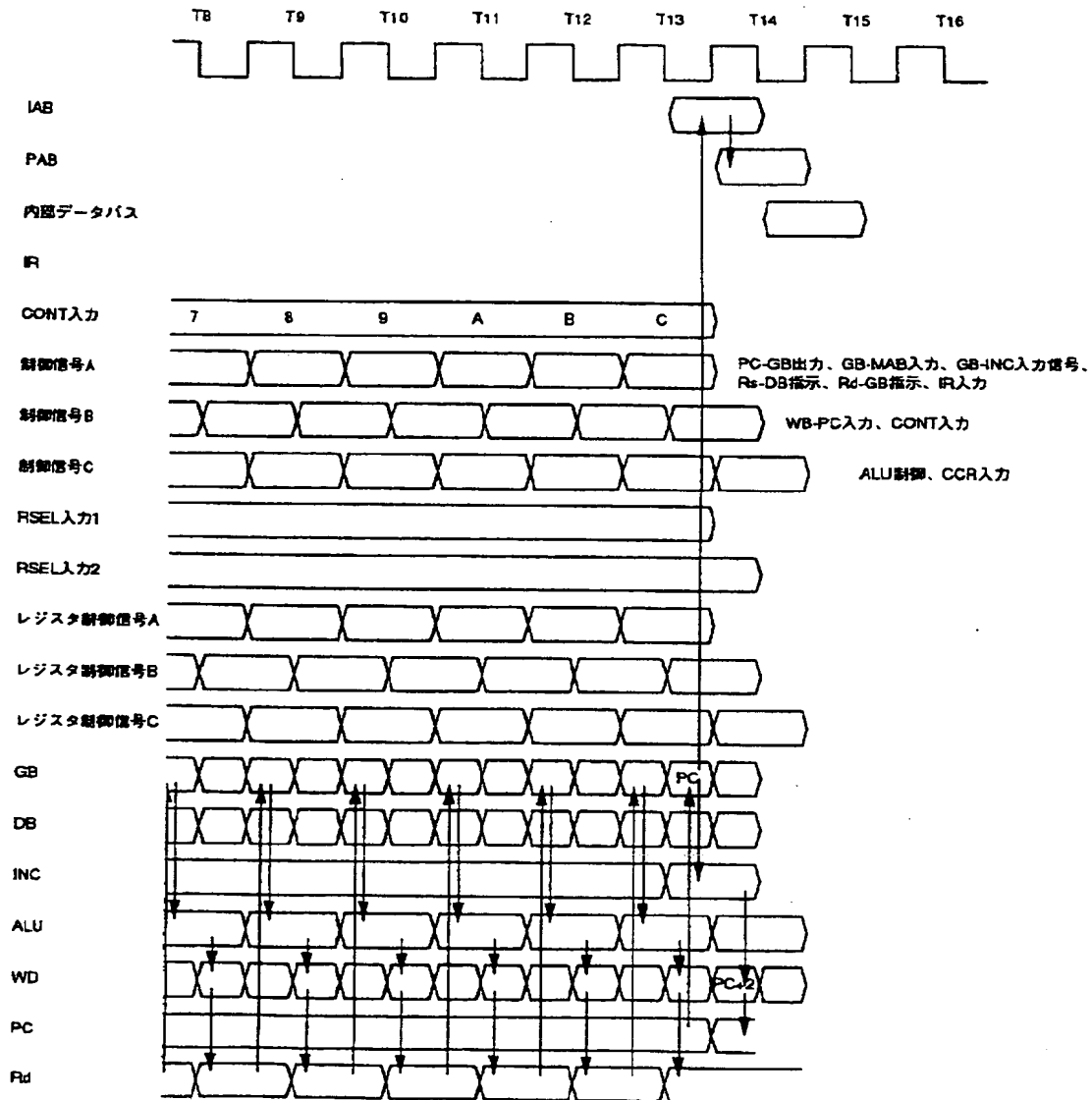
【図 47】

図 47



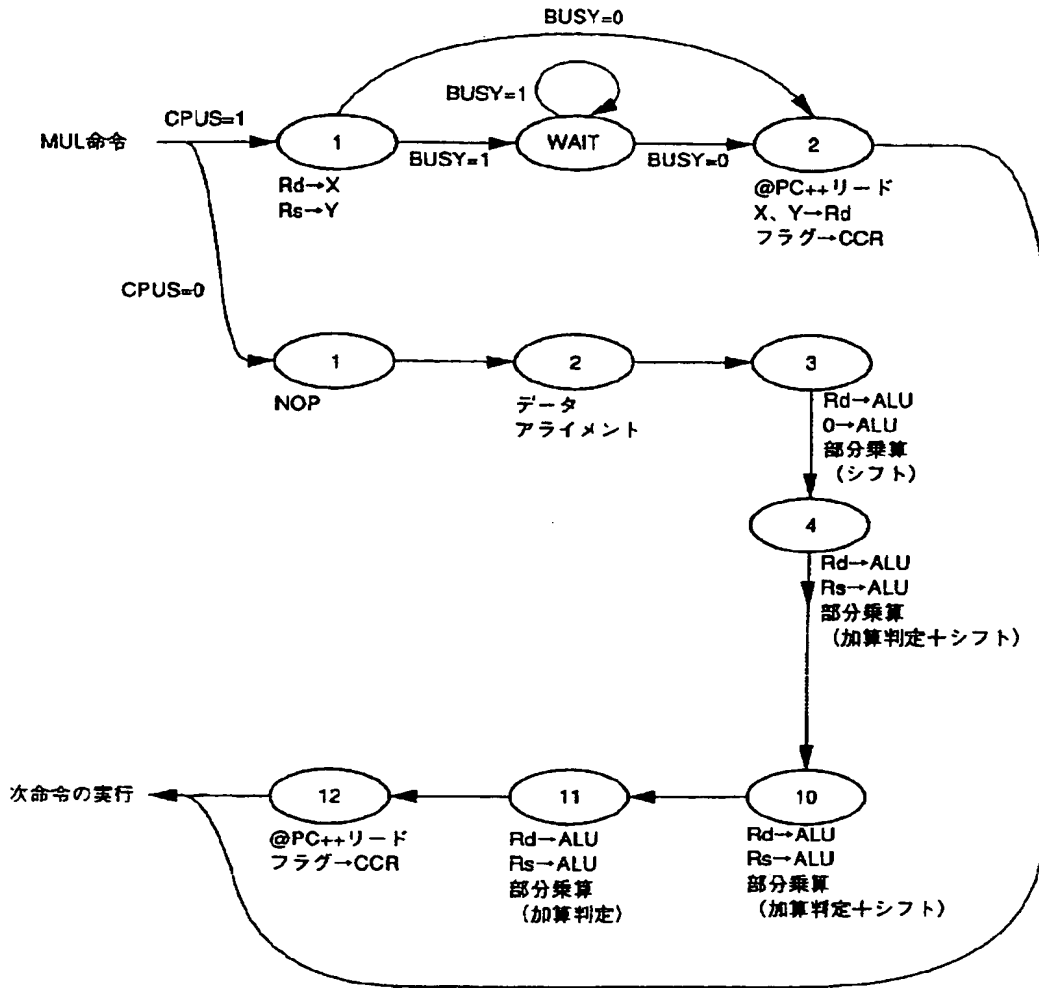
【図26】

図26



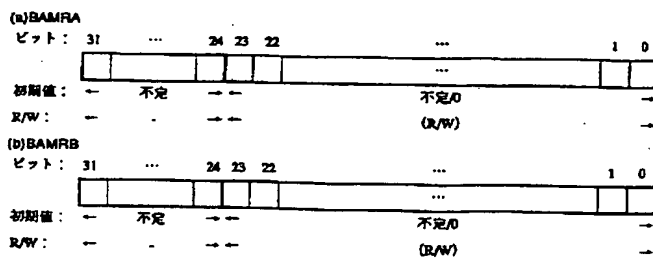
【図 27】

図 27



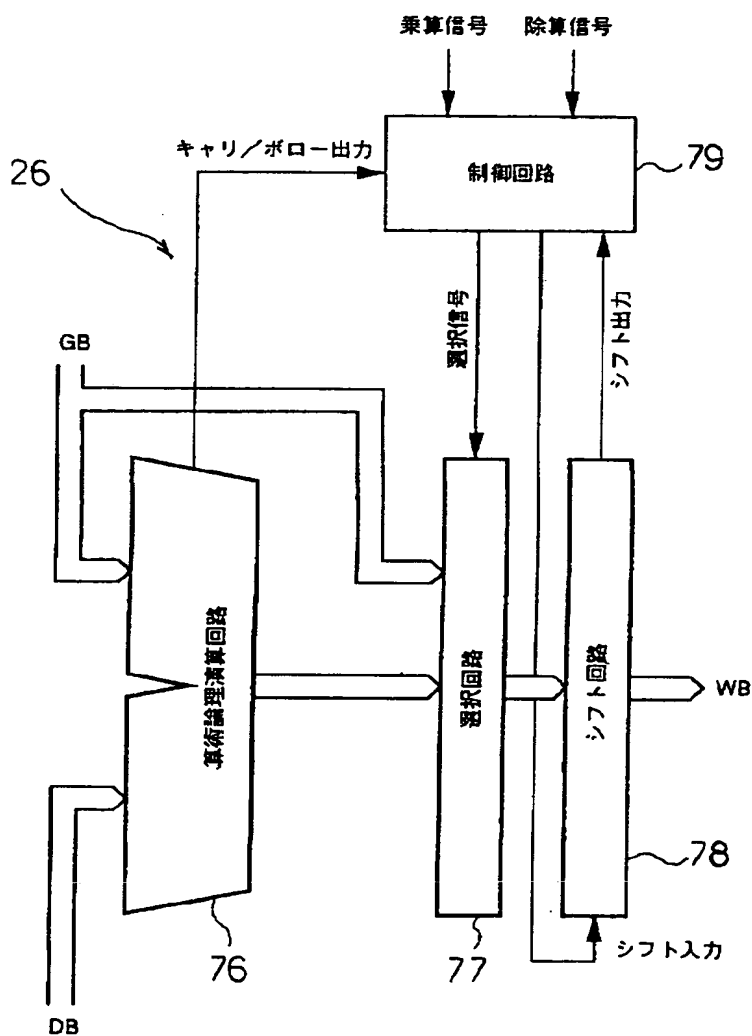
【図 48】

図 48



【図 28】

図 28



【図 36】

図 36

```

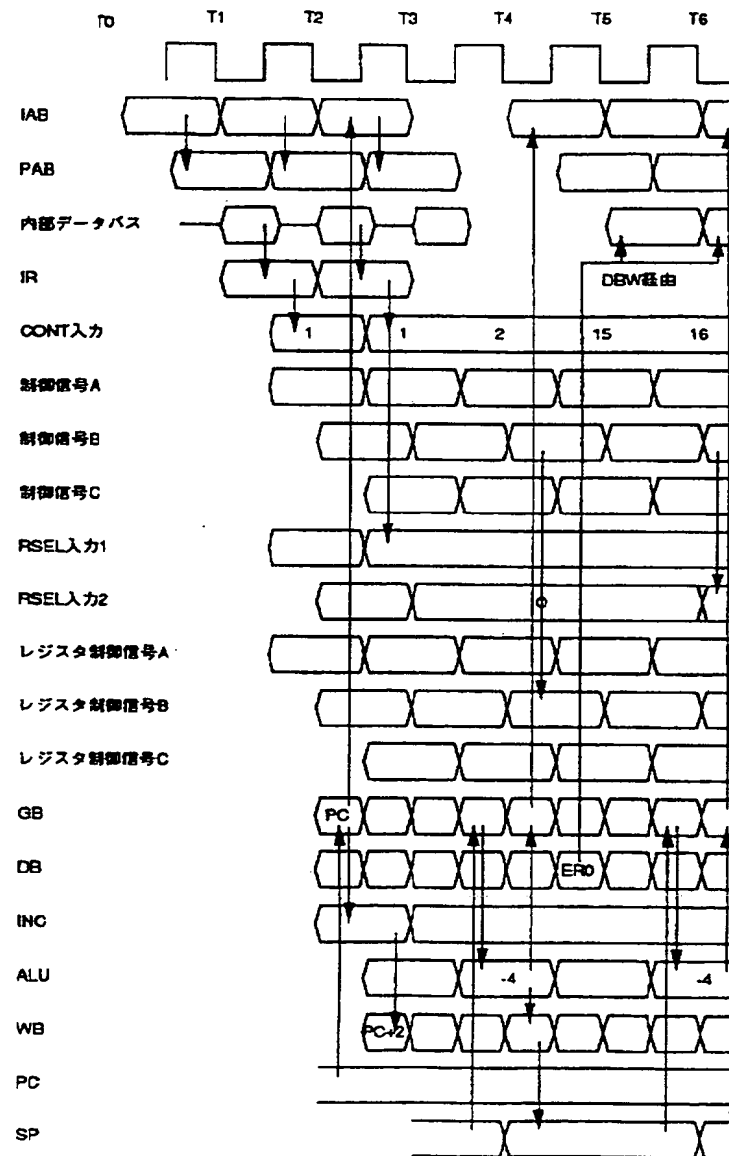
270:  #undef NextRecord
271:  }
00000290 01206D76      LDM L    @SP+, (ER4-ER6)
00000294 01106D73      LDM L    @SP+, (ER2-ER3)
00000298 5470      RTS

272:
273:
274:
275:
276:
277:
278:  void Proc3(register RecordPtr @PtrParOut)
279:  _Proc3:      : function: Proc3
280:  {
281:      PUSH L    ER5
282:      MOV L     @PtrGlb:32, ER1
283:      MOV L     ER0, ER1
284:      #ifdef DEEBOG
285:      printf("Proc3 executed %n");
286:      #endif
287:      if (PtrGlb=NULL)
288:      MOV L     @ER1, ER0
289:      BEQ      L190:8
290:      @PtrParOut=PtrGlb->PtrComp;
291:      MOV L     @ER1, ER0
292:      MOV L     @ER0, ER0
293:      MOV L     ER0, @ER5
294:      BRA      L191:8
295:  L190:
296:  else
297:      IntGlb=100;
298:      MOV W     $100:16, R0
299:      MOV W     R0, @_IntGlb:32
300:      <00000000>
301:  L191:
302:      MOV L     @ER1, ER5
303:      ADD L     $6:32, ER5
304:      MOV W     @_IntGlb:32, R0
305:      ADD W     $12:16, R0
306:      MOV W     R0, @ER5
307:      Proc7(10, IntGlb, &PtrGlb->IntComp);
308:  }
00000304 01006D76      POP L    ER5
00000308 5470      RTS
309:
310:

```

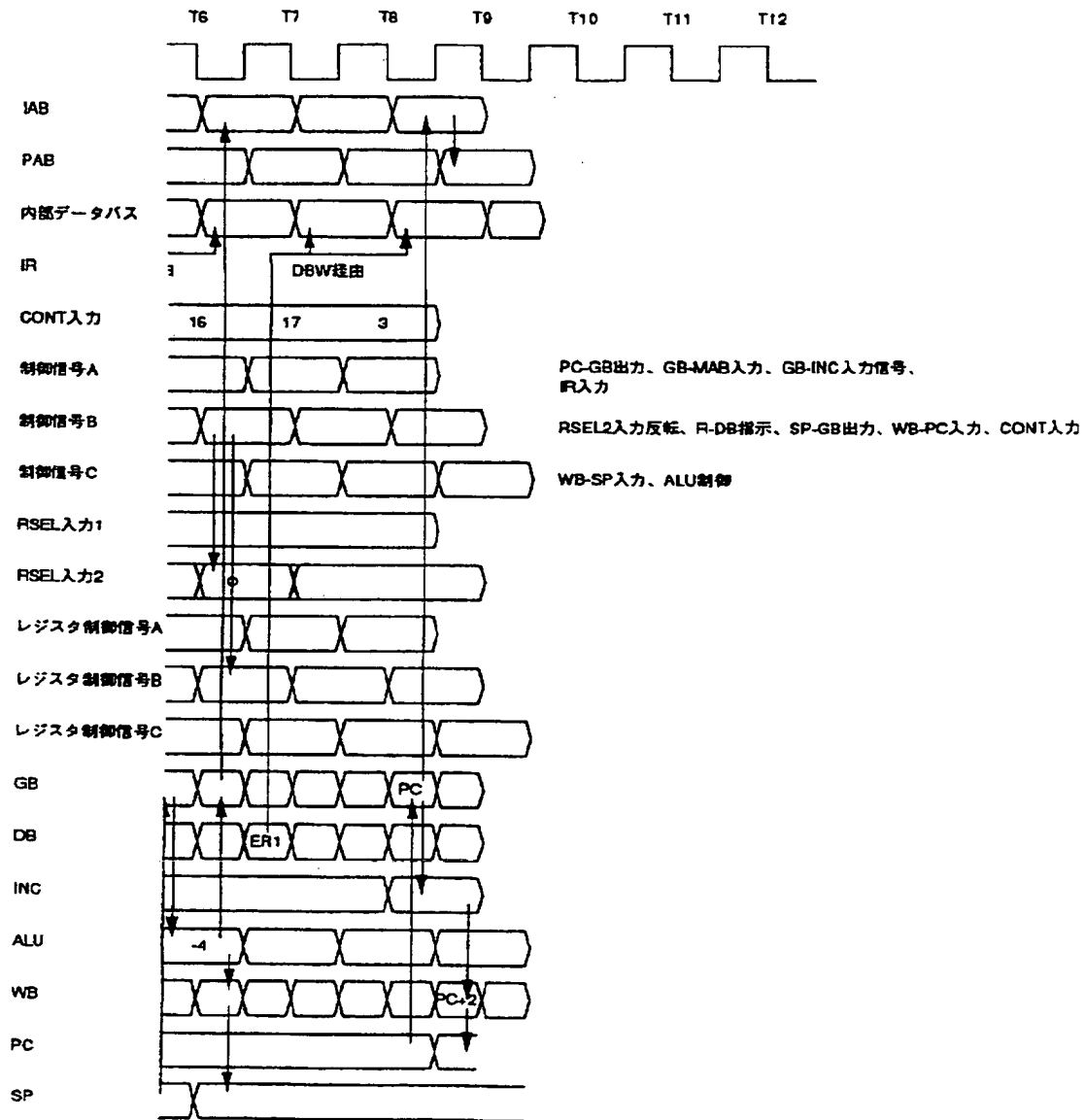
【図29】

図 29



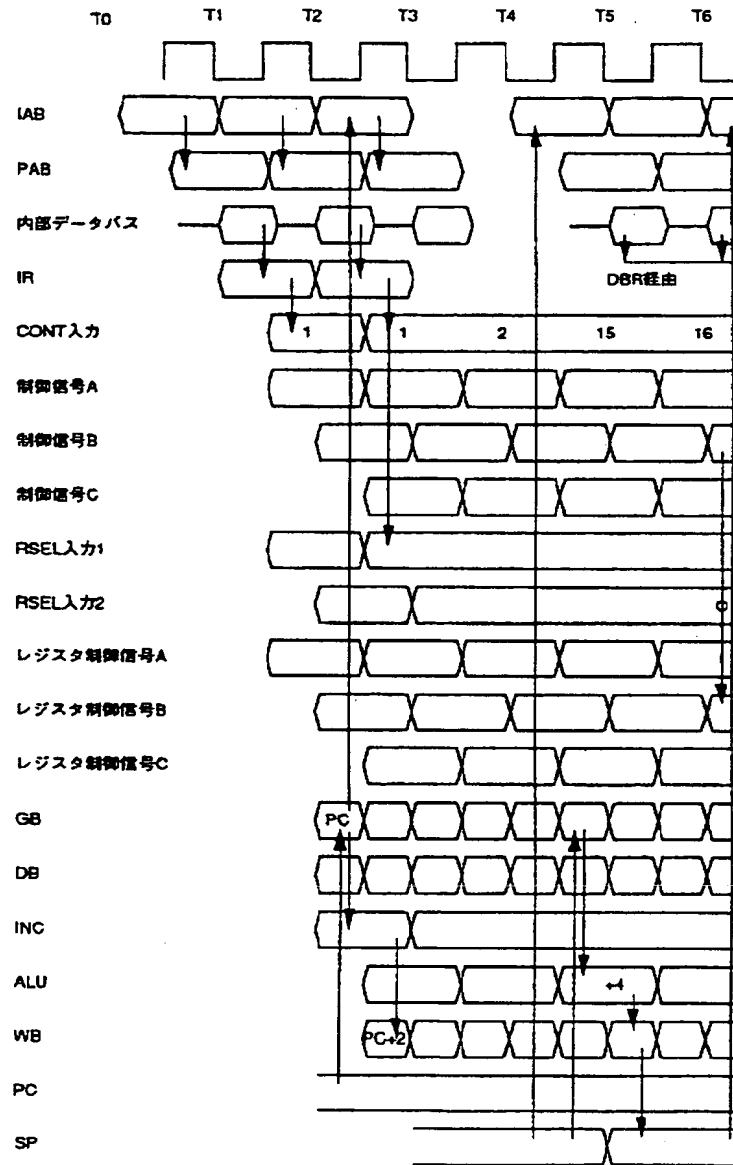
【図 30】

図 30



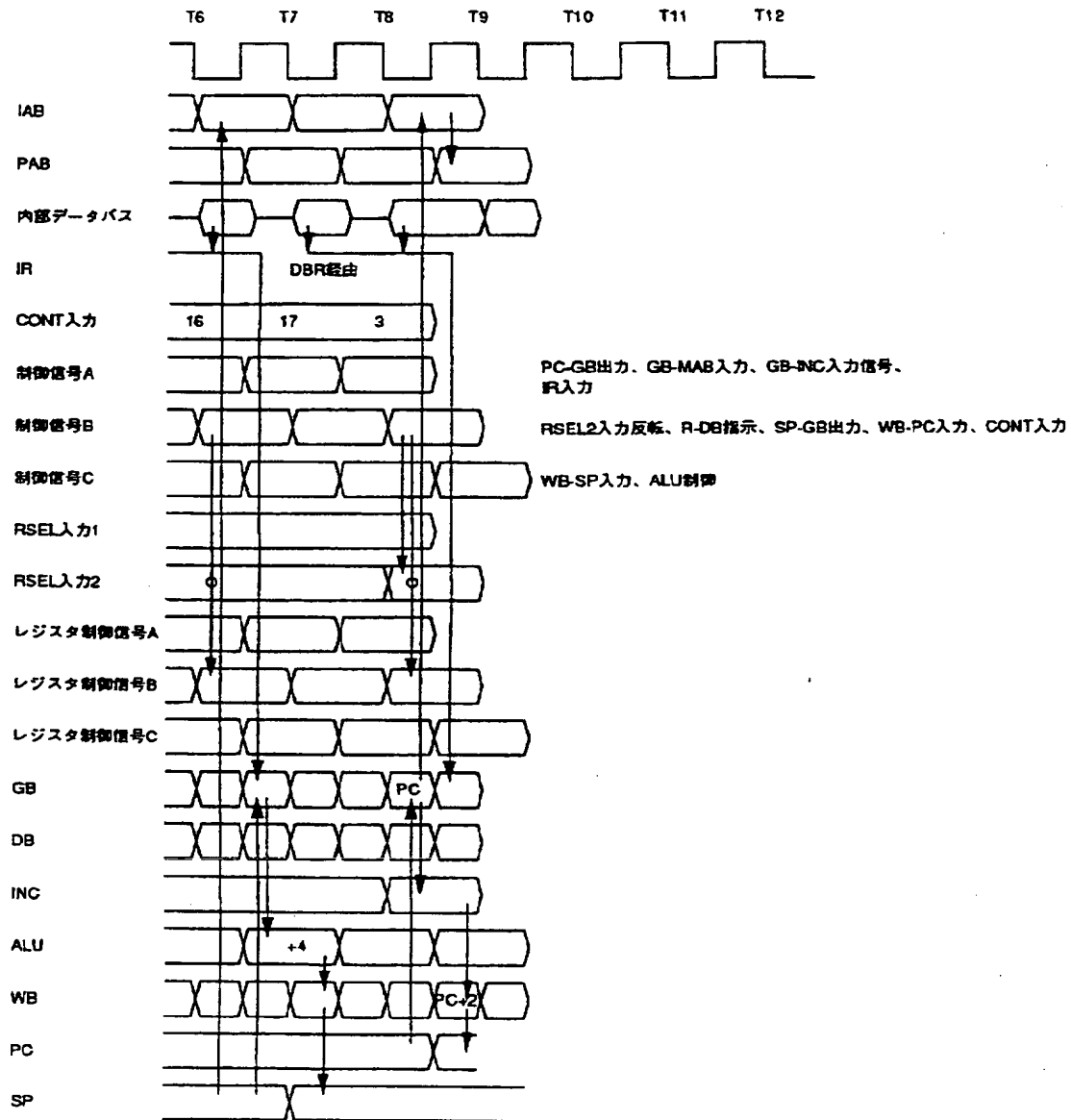
【図31】

図 3 1



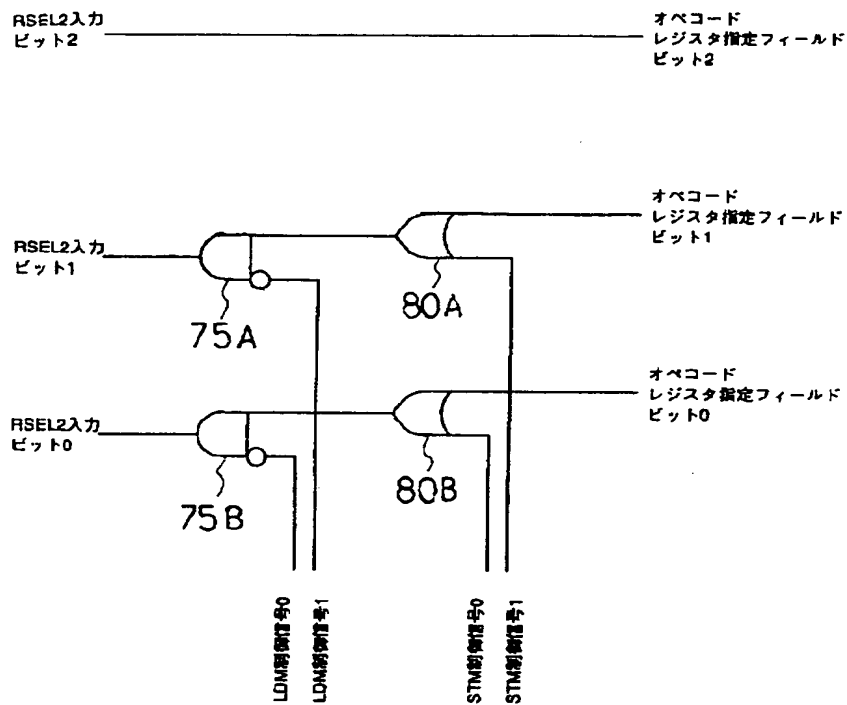
【図 3 2】

図 3 2



【図33】

図 3 3



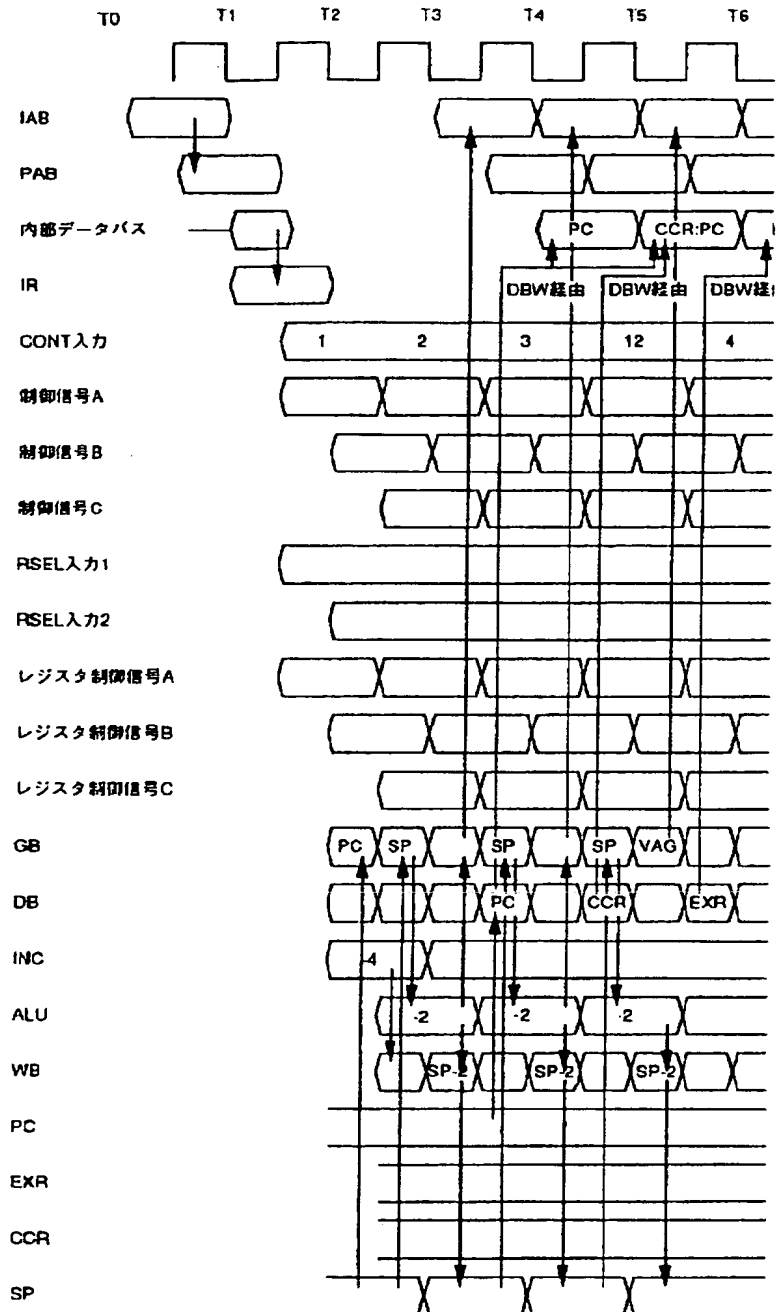
【図35】

図35

SCT	OFFSET	CODE	C LABEL	INSTRUCTION	OPERAND	COMMENT
			248:			
			249:	void Proc1(REQ RecordPtr PtrParIn)		
000001D6			_Proc1:			: function: Proc1
000001D6	01106DF2			STM.L	(ER2-ER3), @-SP	
000001DA	01206DF4			STM.L	(ER4-ER6), @-SP	
		250:	{			
000001DE	0F86			MOV.L	ER0, ER6	
		251:	#define	NextRecord	(*(PtrParIn->PtrComp))	
		252:				
		253:	#ifdef	DEBUG		
		254:	printf	("Proc1 executed. Yn");		
		255:	#endif			
		256:	structassign	((NextRecord), (*PtrGlb));		
000001E0	01006961			MOV.L	@ER6, ER1	
000001E4	01006B20			MOV.L	@_PtrGlb:32, ER0	
						<00000000>
000001EC	7A02000000			MOV.L	#10:32, ER2	
						0A
000001F2			L181:			
000001F2	01006D03			MOV.L	@ER0+, ER3	
000001F6	01006993			MOV.L	ER3, @ER1	
000001FA	0B91			ADDS.L	#4, ER1	
000001FC	1B72			DEC.L	#1, ER2	
000001FE	46F2			BNE	L181:8	
		257:		PtrParIn->IntComp=5;		
		258:		NextRecord.PtrComp=PtrParIn->PtrComp;		
00000210	01006960			MOV.L	@ER6, ER0	
00000214	01006980			MOV.L	ER0, @ER0	
		260:		Proc3(&NextRecord.PtrComp);		
00000218	01006960			MOV.L	@ER6, ER0	
0000021C	5C0000A4			BSR	_Proc3:16	
		261:		if(NextRecord.Discr==10ent1)		
00000220	01006960			MOV.L	@ER6, ER0	
00000224	6E090004			MOV.B	@(4:16, ER0), R1L	
00000228	464C			BNE	L182:8	
		262:	{			

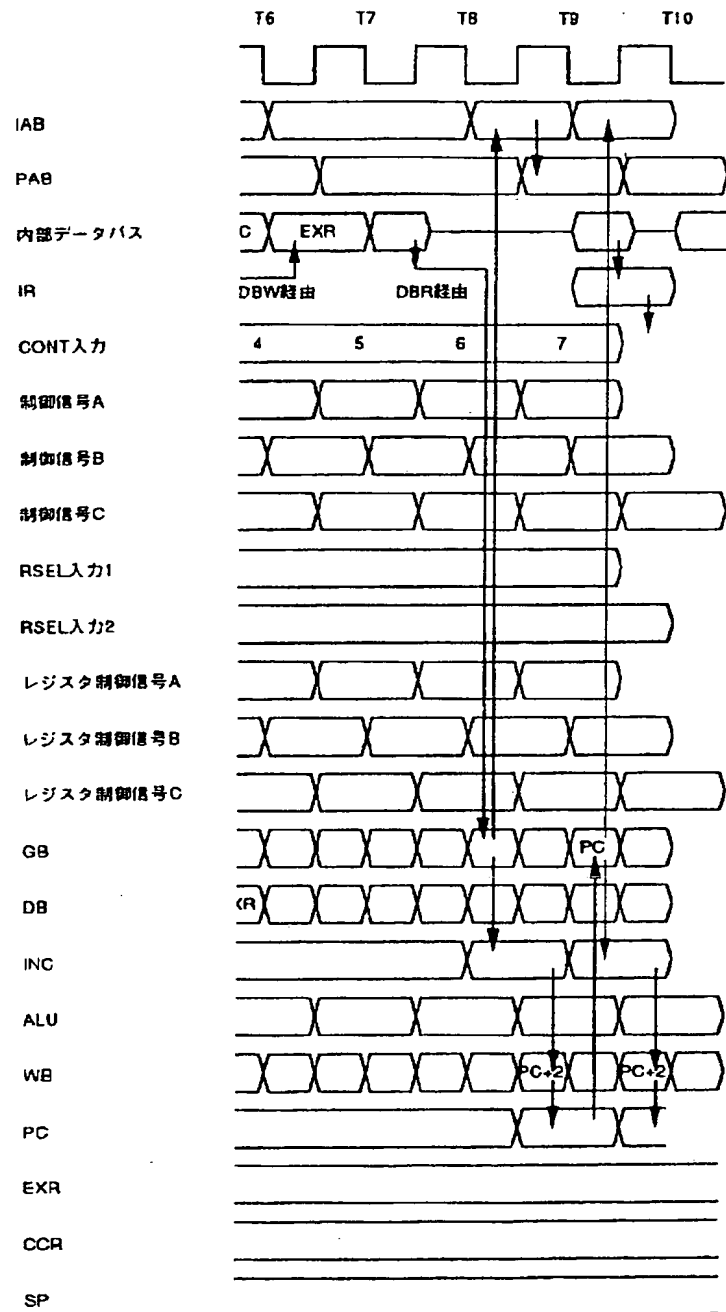
【図 3 7】

図 3 7



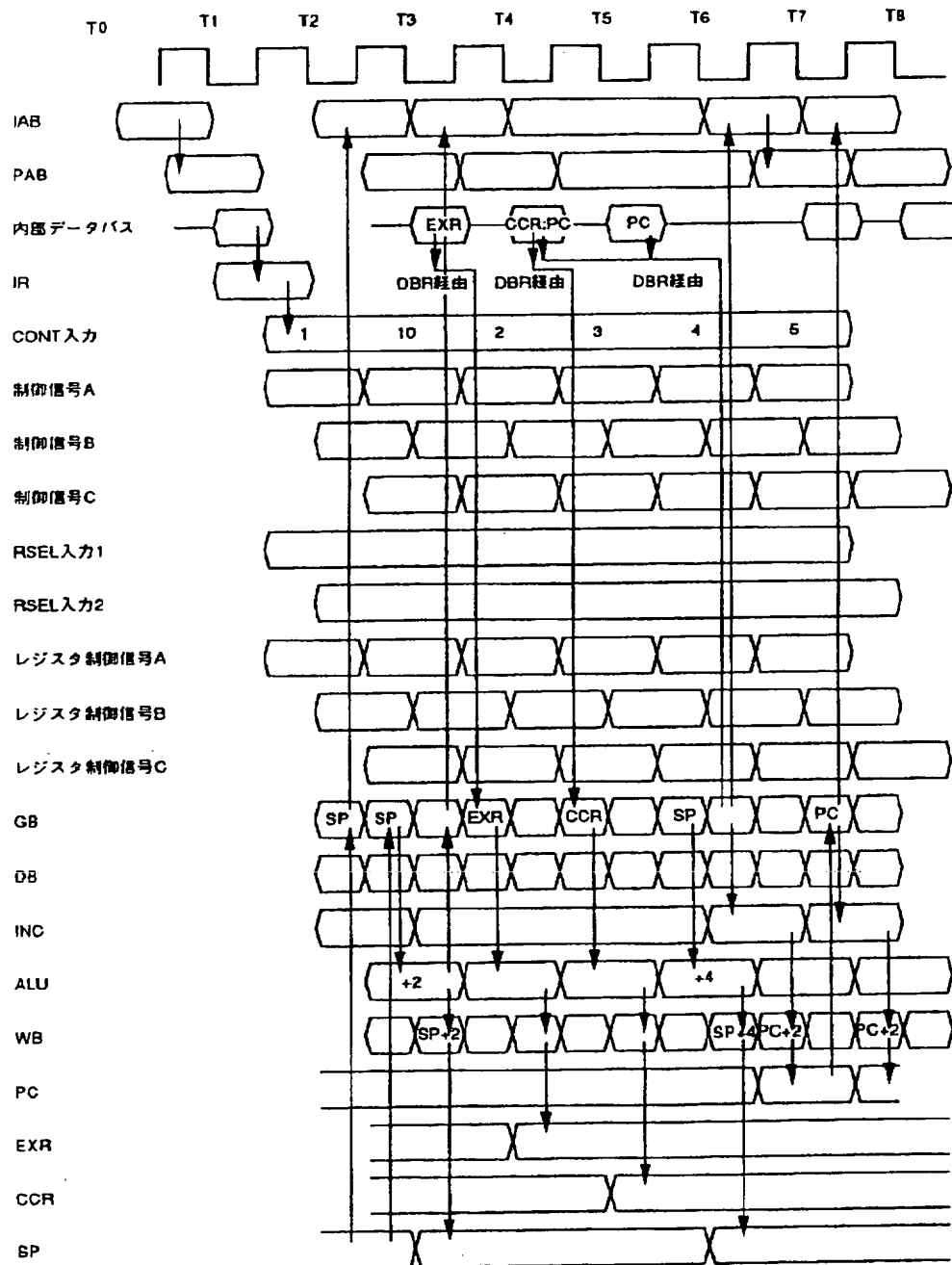
【図38】

図 38



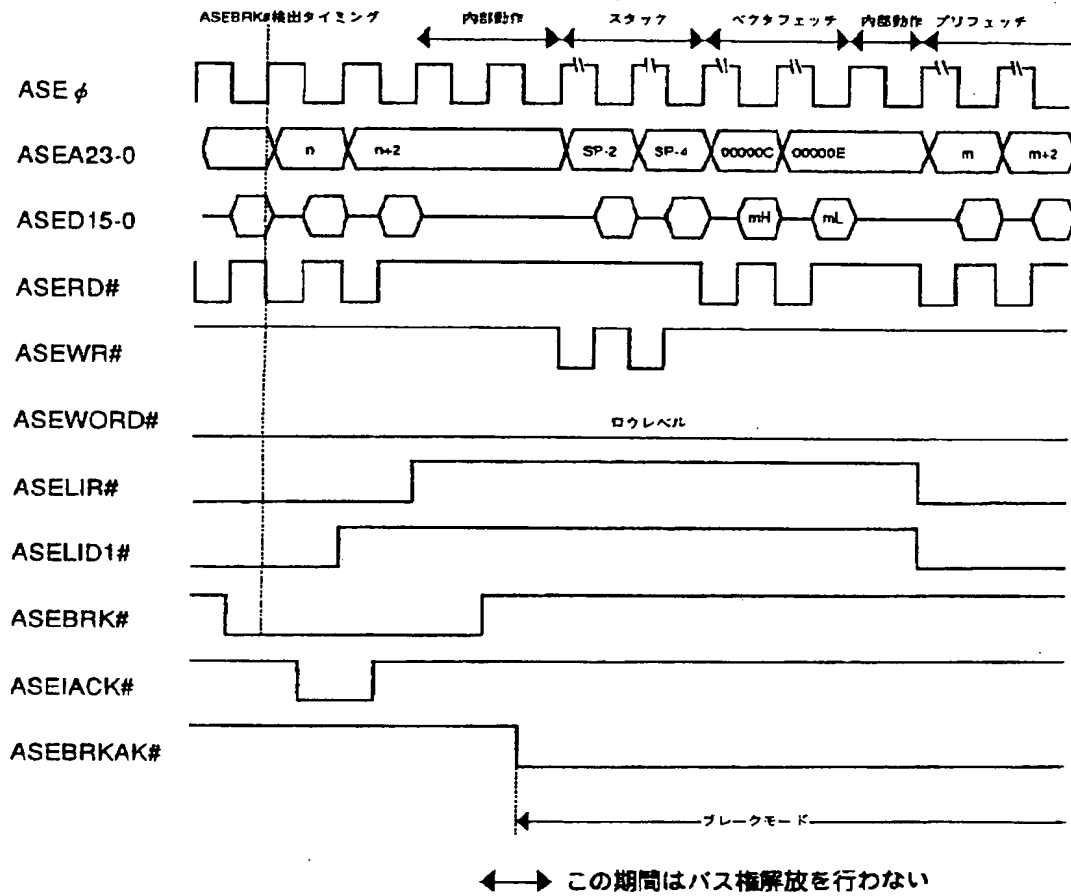
【図42】

図 4 2



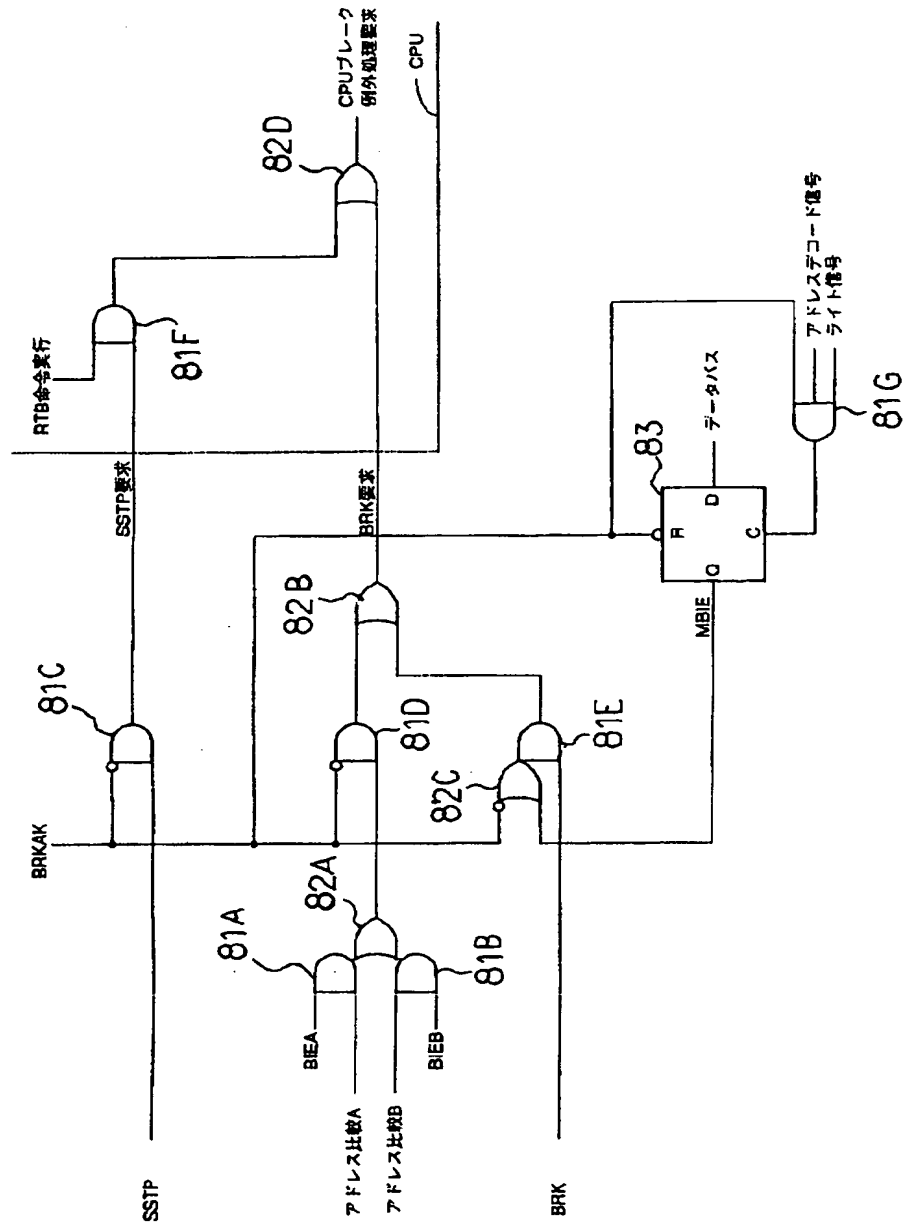
【図 5 1】

図 5 1



【図 52】

図 52



フロントページの続き

(72)発明者 梶原 久志
茨城県日立市幸町3丁目2番1号 日立エ
ンジニアリング株式会社内

(72)発明者 宇枝 晋
茨城県日立市幸町3丁目2番1号 日立エ
ンジニアリング株式会社内

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.